Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακή Διατριβή
στα Πληροφοριακά Συστήματα

Μηχανισμοί Μάθησης και η Χρήση τους στην Επικύρωση της Ποιότητας Νέων Παιγνίων

Σπύρος Γκεζερλής

Επιβλέπων Καθηγητής
Δημήτριος Καλλές

Αύγουστος 2012

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μηχανισμοί Μάθησης και η Χρήση τους στην Επικύρωση της Ποιότητας Νέων Παιγνίων

Σπύρος Γκεζερλής

Επιβλέπων Καθηγητής

Δημήτριος Καλλές

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε
προς μερική εκπλήρωση των απαιτήσεων για απόκτηση

μεταπτυχιακού τίτλου σπουδών
στα Πληροφοριακά Συστήματα

από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών
του Ανοικτού Πανεπιστημίου Κύπρου

Αύγουστος 2012

Περίληψη

Το αντικείμενο είναι η αξιοποίηση υπάρχουσας τεχνογνωσίας και τεχνολογίας πάνω στην χρήση της ενισχυτικής μάθησης για την αυτόματη ανακάλυψη τακτικής σε ένα παιχνίδι, ώστε ο μηχανισμός μάθησης να χρησιμοποιηθεί σε νέα, υπό σχεδίαση παιχνίδια.

Στην προτεινόμενη εργασία επιχειρούμε να σχεδιάσουμε ένα νέο παιχνίδι (ανταγωνιστικό, μεταξύ δυο παιχτών) και να επικυρώσουμε ορισμένες ιδιότητες του (πχ ισορροπία αποτελεσμάτων με βάση το τυχαίο παιχνίδι, ταχύτητα μηχανικής μάθησης με βάση την πολυπλοκότητα του παιχνιδιού, κλπ) και τη δυνατότητα του υπολογιστή να μαθαίνει από παρατήρηση παικτών (υπολογιστικών η μη).

Χρησιμοποιούμε τις υπάρχουσες υλοποιήσεις και επεκτεινόμαστε αξιοποιώντας μόνο τη συνιστώσα της ενισχυτικής μάθησης.

Summary

Machine Learning Mechanisms and their Use in Validating the Quality of Novel Games

Our objective is to use existing knowhow with reinforcement learning and its implementation technology to automatically discover tactics in a game, so that the learning mechanisms can be used in novel games, currently uder developement.

In this thesis we are trying to design a new zero-sum between two players and validate its properties (e.g fairness based on a random game, learning speed based on its complexity etc.) and the computers ability to learn by observing other players (humans or not).

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την γυναίκα μου Αθανασία, τις κόρες μου Μυρσίνη και Αριάδνη και φυσικά την μητέρα μου Μαρία.

Τέλος, τον καθηγητή μου Δημήτρη Καλλέ, καθώς από το πρώτο έως και το τελευταίο έτος υπήρξε πάντα αρωγός στην ολοκλήρωση του Μεταπτυχιακού μου.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Στην παρούσα μεταπτυχιακή διατριβή πραγματοποιείται κατασκευή εργαλείων, βιβλιοθηκών, γραφικού Παιχνιδιού και Πράκτορα Τεχνητής Νοημοσύνης με χρήση της γλώσσας προγραμματισμού Python, μιας ώριμης, αξιόπιστης και επεκτάσιμης αντικειμενοστραφούς γλώσσας. Οι βιβλιοθήκες που υπάρχουν διαθέσιμες τυποποιούν την ανάπτυξη ακολουθώντας συγκεκριμένα σημεία αναφοράς όπως το Pygame, Matplotlib, Scipy, Numpy κλπ.

Η υλοποίηση της εργασίας θα είναι ένα παιχνίδι 2 παικτών με τα ακόλουθους τρόπους εκτέλεσης: άνθρωπος-άνθρωπος, άνθρωπος-μηχανή, μηχανή-μηχανή. Το περιβάλλον είναι μια δεξαμενή χωρισμένη από μια γέφυρα πολλαπλών βυθιζόμενων και ανυψωμένων πλατφορμών (παραλλαγή Διώρυγας Παναμά) με στόχο την καταπόντιση του αντιπάλου στο νερό.

Οι στρατηγικές, τα σενάρια δοκιμών και η λογική θα είναι πλήρως παραμετροποιήσιμα από τον χρήστη, και ακολουθεί τις αρχές της Ενισχυτικής Μάθησης.

1.1 Χρονοδιάγραμμα

Το έργο χωρίζεται σε 3 τμήματα - ο αρχικός προγραμματισμός εργασίας βάσει αυτών των: Κατασκευή Παιχνιδιού, Δημιουργία Πράκτορα, Εκτέλεση σεναρίων.

1. Υλοποίηση απλού και παραμετροποιήσιμου ως προς τους κανόνες παιχνιδιού άνθρωπος-άνθρωπος
   3 μήνες (Νοε-Δεκ - Ιαν)
2. Επόμενο βήμα υλοποίηση Πράκτορα Ενισχυτικής Μάθησης
   4 μήνες (Ιαν-Φεβ-Μάρ-Απρ)
3. Αποτύπωση δοκιμών και αποτελεσμάτων αυτόματης ανακάλυψης τακτικής και αξιολόγησης βαθμού ενδιαφέροντος του παιχνιδιού από απλούς χρήστες
   4 μήνες (Μάρ-Απρ-Μάι-Ιουν)

1.2 Εργαλεία

Για την ανάπτυξη χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python. Πρόκειται για μια ώριμη, αξιόπιστη και επεκτάσιμη αντικειμενοστραφής γλώσσα. Οι βιβλιοθήκες που υπάρχουν διαθέσιμες τυποποιούν την ανάπτυξη ακολουθώντας συγκεκριμένα σημεία αναφοράς όπως το Pygame, Matplotlib, Scipy, Numpy, Pygraphviz, Networkx, PyDot.

Το ολοκληρωμένο εργαλείο ανάπτυξης που επιλέχθηκε είναι το Aptana Studio 3 με χρήση της βιβλιοθήκης PyDev.

Επίσης αξίζει να σημειωθεί πως έγινε εκτενής προσπάθεια ενσωμάτωσης πρώτα της βιβλιοθήκης RL-Glue καθώς και PyBrain. Οι δύο αυτές βιβλιοθήκες θεωρούνται αξιόλογες προσπάθειες στην προγραμματιστική τυποποίηση της Ενισχυτικής Μάθησης. Οι λόγοι που δεν επελέγησαν είναι:

1. Μεγάλος χρόνος εκμάθησης
2. Δυσκολία ενσωμάτωσης στο σύστημα

1.3 Αποθετήριο

Ως πυρήνας της παρούσας μεταπτυχιακής διατριβής θεωρείται το αποθετήριο έργου:

http://code.google.com/p/rltankattack/

Η συγκεκριμένη μέθοδος “Online Project Hosting” έδινε την δυνατότητα συνεχούς παρακολούθησης από τον Επιβλέποντα για την πορεία του έργου. Οι προτυποποιήσεις “ανέβαιναν” στον χώρο αυτό και υπήρχε τάχιστη συνεργασία και σχόλια πάνω στα επόμενα βήματα της Διατριβής.

Η παρούσα υπηρεσία της Google, το Google Code, παρέχει μια πληθώρα εργαλείων αποθήκευσης, ορισμού εκδόσεων, wiki σελίδων κλπ για πιο ολοκληρωμένη οργάνωση των επιμέρους τμημάτων ενός έργου λογισμικού.

Το έργο ονομάστηκε RLTankAttack (επίθεση με χρήση δεξαμενών νερού χρήσει RL). Αναφορά στο ίδιο το παιχνίδι ακολουθεί στο κεφάλαιο 2 “Σχεδίαση”.

Στον παραπάνω ιστοχώρο καταχωρούνται από την αρχή της ανάπτυξης της εργασίας τα εξής 3 βασικά στοιχεία:

1. Κώδικας εφαρμογής και εκτελέσιμα
1. Κείμενα, διαγράμματα περιγραφής και μελέτες ανάπτυξης
2. Σελίδες Wiki για γρήγορη περιήγηση και περιγραφή online.

1.3.1 Subversion (SVN)

Στην καρδιά του έργου RLTankAttack βρίσκεται το SVN Version Control System, ώστε κώδικας, διαγράμματα, εικόνες και κείμενα να διαθέτουν εκδόσεις εξέλιξης. Το SVN είναι μια ισχυρή μέθοδος ώστε να δουλεύουν πολλοί χρήστες σε κοινό έργο, να αποθηκεύουν (commit) τις αλλαγές τους και σε περίπτωση ταυτόχρονης αποθήκευσης από πολλούς χρήστες να μπορεί να γίνει ανανέωση (update), σύγκριση (diff) ή και συγχώνευση κώδικα (merge).

Κεφάλαιο 2

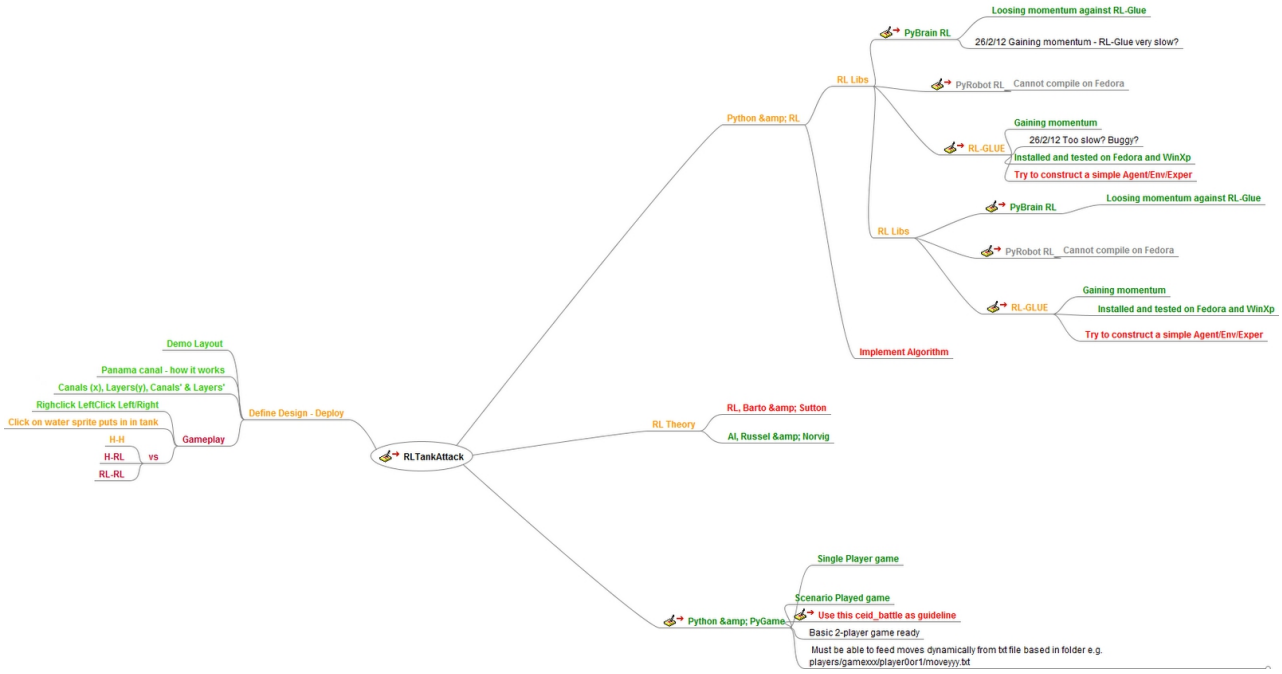
Σχεδίαση

2.1 MindMap

Το όλο έργο ξεκίνησε να σχεδιάζεται με χρήση mindmaps, χάρτη σκέψης που εικονοποιεί πληροφορίες, σκέψεις, ιδέες και ροή ενεργειών, απαιτήσεων και ροή εργασιών σε αρκετά αφαιρετικό επίπεδο.

Τα mindmaps είναι η πρώτη πράξη ενός μηχανικού στην ανάπτυξη ενός έργου όπως φαίνεται και στην εικόνα 2.1: η καταγραφή στο χαρτί σε μορφή βημάτων τα επιμέρους τμήματα του έργου, τις απαιτήσεις και τον σχεδιασμό τους. Οι "φυσαλίδες" σκέψης ορίζουν αφαιρετικά τις εργασίες.

Για την διατριβή ακολουθεί μια αρχική αποτύπωση της καταγραφής αυτής:



Εικόνα 2.1: MindMapping του έργου RLTankAttack, με παρουσίαση των επιμέρους βημάτων.

Το MindMap του έργου ξεκίνησε με τις οντότητες της εμβάθυνσης στην θεωρία της Ενισχυτικής Μάθησης, της Python και των βιβλιοθηκών της. Επίσης στην κατασκευή παιχνιδιών με υπάρχοντα εργαλεία.

Με την παραπάνω ανάπτυξη είναι εφικτός ο διαχωρισμός των εργασιών που έπρεπε να γίνουν εξ αρχής: Της δημιουργίας ενός παιχνιδιού/πειράματος, της δημιουργίας πράκτορα Τεχνητής Νοημοσύνης (ΤΝ) και της εκροής αποτελεσμάτων δοκιμών από την αλληλεπίδραση του πειράματός με τον πράκτορα.

2.2 Ορισμός Εργασιών

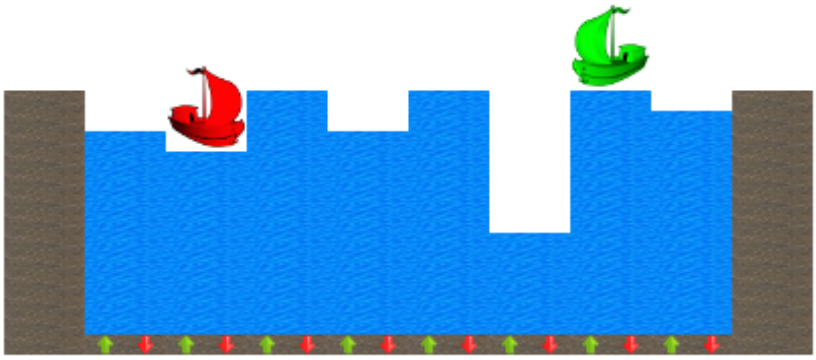
Οι τρεις διακριτοί χώροι ανάπτυξης του πειράματος μας με στόχο την εύρεση αυτόματης ανακάλυψη τακτικής σε ένα παιχνίδι, ώστε ο μηχανισμός μάθησης να χρησιμοποιηθεί σε νέα, υπό σχεδίαση παιχνίδια είναι:

“Παιχνίδι” → “Πράκτορας” → “Δοκιμές”

Με τον διαχωρισμό των παραπάνω τμημάτων ως τομείς ανάπτυξης, είναι διακριτή η εργασία που χρειάζεται το κάθε τι. Το τελικό και αυτονόητο βήμα είναι η σύνθεση των παραπάνω σε ένα ενιαίο σύστημα.

2.2.1 Περιγραφή Παιχνίδιου

Το υπο σχεδίαση παιχνίδι αφορά δεξαμενή χωρισμένη από μια γέφυρα πολλαπλών βυθιζόμενων και ανυψωμένων πλατφορμών (παραλλαγή Διώρυγας Πανομά), 2 αντίπαλα πλοία-παίκτες που ελέγχονται από άνθρωπο ή ΤΝ με στόχο την καταπόντιση του αντιπάλου στο νερό. Το επόμενο υπό επέκταση βήμα θα είναι η κατάκτηση της βάσης του αντιπάλου.



Εικόνα 2.2: Γραφική απεικόνιση πρωτότυπου του παιχνιδιού RLTankAttack. Διακρίνεται το περιβάλλον, οι βαλβίδες εισροής κ' εκροής νερού, τα αντίπαλα πλοία καθώς και η κάθε διακριτή στήλη νερού.

Το γραφικό περιβάλλον του θα πρέπει να διαχωρίζει οπτικά τους παίκτες, το περιβάλλον (νερό, τοίχος κλπ) καθώς και τα χειριστήρια (κίνηση σκάφους, βαλβίδες κίνησης νερού).

2.2.2 Πράκτορας

Ο πράκτορας ΤΝ θα χρησιμοποιεί ενισχυτική μάθηση για την σταδιακή βελτίωση του έναντι στον χρήστη άνθρωπο.

2.2.3 Δοκιμές

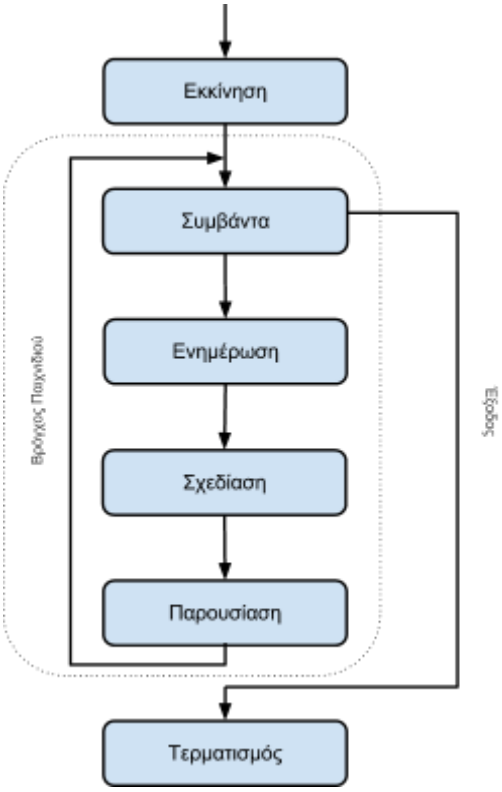
Οι δοκιμές προσπαθούν να παρουσιάσουν δεδομένα που απασχολούν την δημιουργία νέων παιγνίων όπως ταχύτητα μηχανικής μάθησης με βάση την πολυπλοκότητα του παιχνιδιού,

ισορροπία αποτελεσμάτων με βάση το τυχαίο παιχνίδι και τη δυνατότητα του υπολογιστή να μαθαίνει από παρατήρηση παικτών.

2.3 Παιχνίδι

2.3.1 Ορισμός παιχνιδιού και κύκλος ζωής

Το παιχνίδι με την ονομασία RLTankAttack περιγράφεται ακολούθως, καθώς γίνεται επίσης και παρουσίαση του στόχου και αντικειμένου του. Τέλος γίνεται ορισμός της φύσης των παικτών. Ο απλός κύκλος ζωής ενός παιχνιδιού έχει ως εξής:



Διάγραμμα 2.1: Τυπικός βρόγχος παιχνιδιού - εκκίνηση, αναμονή εντολών, ανανέωση και τερματισμός.

Το παραπάνω τυπικό σενάριο ζωής ενός παιχνιδιού ορίζει όλο τον σχεδιασμό και αλληλεπίδραση παίκτη, την αποτύπωση στην οθόνη και τον τερματισμό του κύκλου του παιχνιδιού. Πάνω σε αυτό το διάγραμμα έχει γίνει η ανάπτυξη του RLTankAttack.

2.3.1.1 Περιγραφή

Το RLTankAttack είναι ένα διαδραστικό γραφικό παιχνίδι αντίταξης δυο (2) παικτών. Το παιχνίδι πρέπει να εκτελείται μέσω της γλώσσας προγραμματισμού Python και των βοηθητικών βιβλιοθηκών της σε πληθώρα συστημάτων όπως GNU/Linux, MS Windows, Mac OS κλπ. Η φορητότητα του στο μέλλον θα μπορεί να μεταφερθεί και σε έξυπνα τηλέφωνα καθώς έγιναν δοκιμές πρωτότυπων σε Android συσκευές με επιτυχία.

2.3.1.2 Αντικείμενο

Στόχος του κάθε παίκτη είναι η νίκη έμμεσα με καταπόντιση του αντιπάλου του ή σε κενό είτε σε στήλη νερού, ή άμεσα μέσω της κατάκτησης του αντίπαλου καρνάγιου.

2.3.1.3 Παίκτες

Η φύση των παικτών-αντιπάλων μπορεί να είναι ένα από τα ζευγάρια:

1. Άνθρωπος-Άνθρωπος
2. Άνθρωπος-Μηχανή
3. Μηχανή-Μηχανή

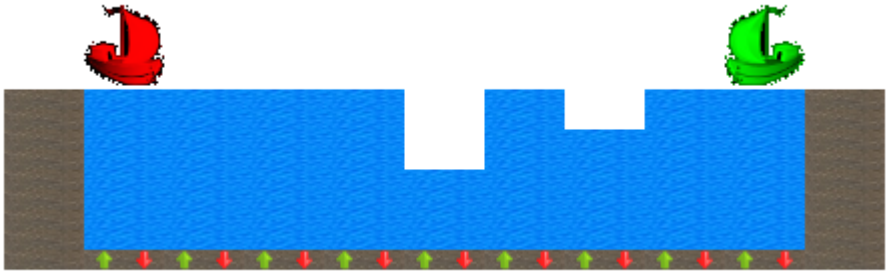
2.3.2 Οντότητες παιχνιδιού

2.3.2.0 Το πλέγμα και οι πίνακες

Το πλέγμα του παιχνιδιού αποτελείται από τρεις ιδίων διαστάσεων ΠxΥ (Πλάτος x Ύψος) ή MxN πίνακες:

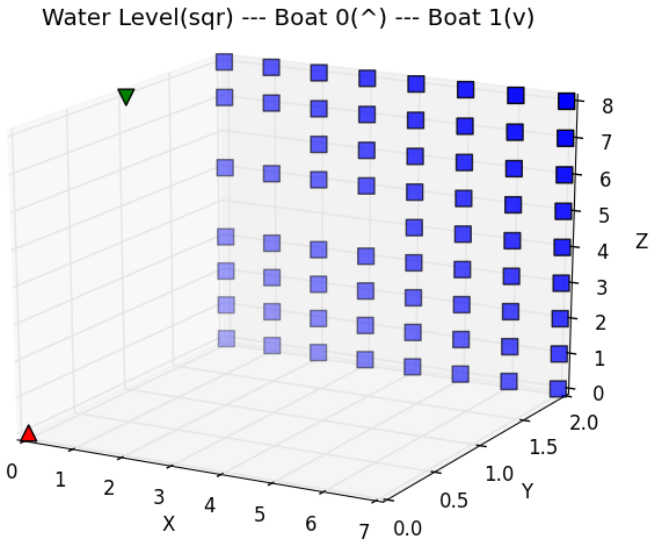
1. Πίνακας νερού → WL_{MxN} (Water Level MxN)
2. Πίνακας πλοίου 0 → B^0_{MxN} (Boat 0 MxN)
3. Πίνακας πλοίου 1 → B^1_{MxN} (Boat 1 MxN)

Ο WL_{MxN} περιέχει την τιμή 0 για ανυπαρξία νερού και 1 για παρουσία αυτού. Οι B^0_{MxN} και B^1_{MxN} έχουν την τιμή 0 σε όλες τις συντεταγμένες και 1 μόνο στο σημείο που βρίσκονται.



Εικόνα 2.3: Αποτύπωση διεπαφής του παιχνιδιού μια τυχαία στιγμή: Οι στήλες 5 και 7 έχουν στραγγίσει κατά 4 και 2 επίπεδα νερού.

Το πλέγμα ταύτισης της παραπάνω κατάστασης μπορεί να οπτικοποιηθεί από την ακόλουθη εικόνα:



Εικόνα 2.4: Πλέγμα ταύτισης των τριων πινάκων B^0_{MxN}, B^1_{MxN} & WL_{MxN}

2.3.2.1 Περιβάλλον

Ως περιβάλλον του παιχνιδιού αναφέρονται τα στοιχεία τα οποία δεν αλληλεπιδρούν με ενέργειες του χρήστη. Διακρίνονται σε διακοσμητικά και ενημερωτικά. Τα διακοσμητικά είναι για παράδειγμα η γη που περιβάλλει το νερό, ήτοι η δεξαμενή που κρατά όλες τις

στήλες. Ενημερωτικά είναι τα βέλη που δείχνουν την σειρά του παίκτη και τα παράθυρα νίκης ή ήττας.

2.3.2.2 Πλοίο

Το πλοίο είναι μια κλάση αντικειμένου που μπορεί να είναι άνθρωπος ή μηχανή. Έχει την δυνατότητα να κινείται αριστερά και δεξιά με χρήση του δεξιού ή αριστερού κουμπιού του ποντικιού. Η θέση του βρίσκεται μέσα σε ένα πίνακα πλοίων διαστάσεων ΠxΥ (Πλάτος x Ύψος νερού μέσα στην δεξαμενή). Με αυτό τον πίνακα γίνεται άμεσος εντοπισμός της τοποθεσίας του στο πλέγμα του παιχνιδιού. Ο πίνακας αυτός είναι B^0_{MxN} ή B^1_{MxN}.

2.3.2.3 Νερό

Η οντότητα νερό βρίσκεται σε ΠxΥ (Πλάτος x Ύψος) μέσα πάντα στην δεξαμενή μας. Το που βρίσκεται νερό στον WL_{MxN} ορίζεται με 0 ή 1.

2.3.2.4 Στήλη νερού

Πρόκειται για μια τεχνητή ομάδα οντοτήτων νερού που βρίσκονται στο ίδιο πλάτος m. Η δεξαμενή περιέχει Μ στήλες.

2.3.2.5 Βαλβίδα

Κάθε βαλβίδα μπορεί να πραγματοποιεί εισροή ή εκροή και μόνο αυτό. Σε κάθε στήλη νερού αντιστοιχούν δύο (2) βαλβίδες, άρα σε δεξαμενή Μ στηλών διαθέτουμε 2xΜ βαλβίδες, Μ για εισροή, Μ για εκροή νερού. Με την διοχέτευση ή αποχέτευση αυτού, εφόσον βρίσκεται πάνω στην αντίστοιχη στήλη πλοίο, αυτό μετακινείται πάνω ή κάτω αντίστοιχα.

2.3.3 Διεπαφή

Η διεπαφή με τον χρήστη πρέπει να είναι όσο το δυνατό πιο απλή και αυτονόητη από τον παίκτη, αν είναι δυνατό και από την πρώτη εκτέλεση του παιχνιδιού. Για αυτό το λόγο, τόσο

το γραφικό περιβάλλον όσο και η αλληλεπίδραση πρέπει να είναι μινιμαλιστική και να επικεντρώνεται άμεσα στην διασκέδαση του παίκτη.

2.3.4 Αλληλεπίδραση και Χειρισμός

Η αλληλεπίδραση του χρήστη με το παιχνίδι γίνεται αποκλειστικά με:

1. συσκευή κατάδειξης όπως το ποντίκι ή οθόνη αφής
2. μέσω των αρχείων καταγραφής (§2.3.6)

Τα ενεργά στοιχεία (Sprites) του παιχνιδιού είναι το πλοίο και οι βαλβίδες. Ο λόγος που σχεδιάστηκε το RLTankAttack να δέχεται με συσκευή κατάδειξης είναι η μελλοντική διαδραστική συμβατότητα, καθώς η τεχνολογία επαφής ανθρώπου μηχανής τείνει προς τις οθόνες αφής. Τα αρχεία καταγραφής είναι ένας απλός και εύκολος τρόπος αλληλεπίδρασης με την ρουτίνα εκτέλεσης του παιχνιδιού στο επίπεδο συμβάντα (διάγραμμα 2.1).

2.3.5 Ορισμοί - Κανόνες - Ενέργειες

2.3.5.1 Ορισμοί

1. Υπάρχουν 2 πλοία αντίπαλοι
2. Η δεξαμενή έχει διαστάσεις MxN (ΠxΥ)
3. Η στάθμη στήλης νερού διαχειρίζεται μέσω των 2 βαλβιδών που της αντιστοιχούν.
4. Ο αριθμός των βαλβίδων είναι διπλάσιος των στηλών νερού.
5. Η δεξαμενή νερού πάνω στο οποίο επιπλέουν τα πλοία αποτελείται από MxN στήλες επί γραμμές νερού.
6. Μια βαλβίδα ελέγχει την στάθμη σε μια στήλη.

2.3.5.2 Κανόνες

1. Το πλοίο 0 ξεκινά αριστερά πάνω και το πλοίο 1 δεξιά πάνω στην δεξαμενή.

2. Κάθε παίκτης εφόσον είναι σειρά του μπορεί να πραγματοποιήσει μέσω των χειριστηρίων μια ενέργεια την φορά.
3. Μια βάρκα μπορεί να βρίσκεται στην ίδια στήλη με μια άλλη.
4. Το πλοίο πρέπει να βρίσκεται πάντα πάνω σε νερό ώστε να είναι λειτουργικό.
5. Κάθε στήλη μπορεί να αδειάζει ή να γεμίζει σε κάθε σειρά παίκτη κατά ένα επίπεδο (0-N).
6. Σκάφος το οποίο μετά από ενέργεια δική του ή άλλου πλοίου δεν διαθέτει νερό στην καρίνα του καταστρέφεται ή καταποντίζεται από ύψος σε νερό.
7. Εφόσον σκάφος περιβάλλεται από νερό ήτοι κινηθεί προς στήλη η οποία περιέχει νερό το πλοίο καταποντίζεται.
8. Εφόσον το σκάφος βρεθεί στην αρχική θέση του αντίπαλου σκάφους πριν από αυτόν, τότε νικάει.

2.3.5.3 Ενέργειες

1. Σκάφος:
    1. Πρόσω
    2. Ανάποδα
2. Βαλβίδα:
    1. Εισροή
    2. Εκροή

Ο έλεγχος σε κάθε περάτωση κίνησης ενός παίκτη αναζητά νικητή και ηττημένο εφόσον η κίνηση αυτή δεν έρχεται σε αντίθεση με τους παραπάνω ορισμούς. Εφόσον υπάρχει παράπτωμα τότε επιστρέφει στην προ κίνησης κατάσταση και αναμένει ενέργεια. Σε αντίθετη περίπτωση και εφόσον όλα έχουν καλώς με την κίνηση, τότε γίνεται σύγκριση των πινάκων B^0_{MxN}, B^1_{MxN} & WL_{MxN} .για τους δύο παίκτες διαδοχικά:

$$\texttt{EXISTS}(B^X_{MxN} \;\&\&\; WL_{MxN}, \texttt{True})$$

Με τον παραπάνω τρόπο έχουμε τιμή 0 σε περίπτωση που το πλοίο X δεν βρίσκεται πάνω σε νερό ή 1 σε περίπτωση που όλα είναι εντάξει και η βάρκα διαθέτει νερό.

2.3.6 Αρχεία καταγραφής

Τα αρχεία καταγραφής αποτυπώνουν:

1. Κινήσεις (πίνακες B^0_{MxN}, B^1_{MxN} & WL_{MxN})
2. Κατάσταση (τερματισμός ή όχι παιχνιδιού)
3. Αμοιβή (-1, 0, 1 η ανταμοιβή του πράκτορα)
4. Σειρά παίκτη (0 ή 1 πλοίο)
5. Τύπος κίνησης (Πλοίο (δεξιά ή αριστερά) ή Στήλη (γέμισμα ή εκροή))
6. Τύπος κίνησης πράκτορα (έξυπνη ή τυχαία για εμπλουτισμό γνώσεων)

Οι τρείς πίνακες B^0_{MxN}, B^1_{MxN} & WL_{MxN} αποτελούν την βασική δομή δεδομένων όλου του παιχνιδιού. Περιέχουν την θέση των πλοίων και την παρουσία νερού. Αυτοί οι πίνακες θα πρέπει να αλληλεπιδρούν τόσο με το παιχνίδι, με τον πράκτορα αλλά και να είναι εύκολα αποθηκεύσιμες με κοινά αποδεκτό και προσβάσιμο τρόπο για να επεξεργάζονται εύκολα και από τις τελικές δοκιμές των πειραμάτων. Η δημιουγία νέου καταλόγου και άρα κίνησης με αποτύπωση των τριων πινάκων έχει αποτέλεσμα την δημιουργία συμβάντος ενέργειας προς το παιχνίδι.

2.4 Πράκτορας

Τα παιχνίδια Η/Υ εκτός από την ψυχαγωγική τους φύση έχουν και το προτέρημα ότι οπτικοποιούν περίπλοκα μαθηματικά προβλήματα.

Πράκτορας λογισμικού είναι ένα αυτόνομο πρόγραμμα ικανό να ελέγχει τις αποφάσεις του στο να δρά βάσει των αντιλήψεων που έχει για περιβάλλον του, με απώτερο σκοπό την επίτευξη στόχου.

Η ενέργειες από την μεριά ενός Πράκτορα πάνω σε αυτό το παιχνίδι, δίνουν στο υπολογιστικό σύστημα να εκμεταλευτεί την δημιουργία λογικής που έχει αποκτήσει από πολύπλοκους αλγόριθμους μάθησης[1].

Ο κύριος λόγος που κάθε παιχνίδι οφείλει να έχει αντίπαλο και μια μηχανή ΤΝ είναι ώστε:

1. να μην είναι απαραίτητη η παρουσία 2ου χρήστη (ακόμη και διαδικτυακά)

[1] Jennings & Wooldridge, 18 Jan 1996, "Software Agents" IEE Review

2. προσαρμογή δυσκολίας ανάλογη με το επίπεδο του αντίπαλου

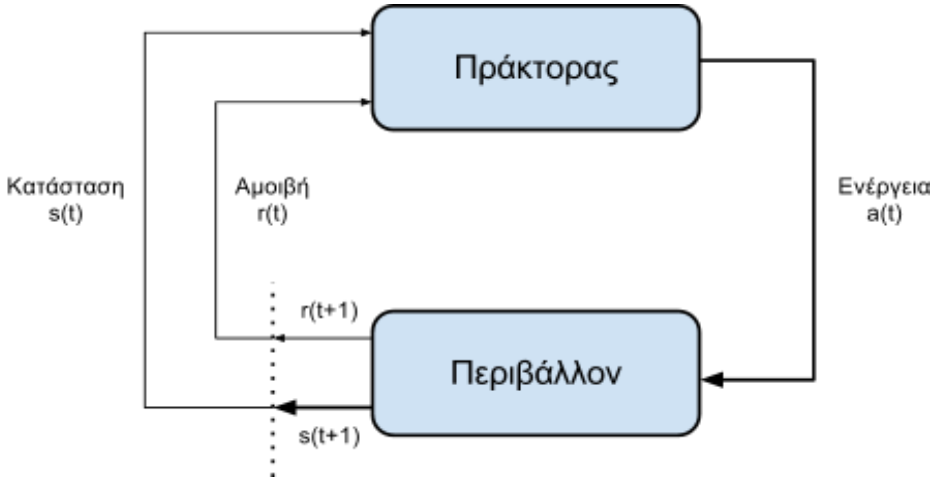
Η τεχνητή νοημοσύνη αποτελεί μια επιστήμη με απέραντες δυνατότητες στον τομέα της επίλυσης προβλημάτων, τομέας που ανήκει και η επίλυση ενός απλού παιγνιδιού. Η επιλεγμένη μέθοδος κατασκευής του πράκτορα μας (Agent) είναι η Ενισχυτική Μάθηση (Reinforcement Learning).

2.4.1 Ενισχυτική Μάθηση

Ο μηχανισμός Ενισχυτικής Μάθησης ή Reinforcement Learning (RL) είναι ο τομέας της μηχανικής μάθησης ώπου ο πράκτορας αισθάνεται το περιβάλλον του, έχει ανάδραση αμοιβής και προσαρμόζει ανάλογα τις πράξεις του. Προσαρμόζει δηλαδή μελλοντικές του ενέργειες βάσει των πεπραγμένων και των αποτελεσμάτων που αυτές είχαν: αν μια ενέργεια στο παρελθόν του επέφερε ζημιά, δεν θα προτιμήσει να την επιλέξει. Αν όμως αυτή τον βοηθά ή και τον οδηγεί σε νίκη, τότε ανεβαίνει στις προτιμήσεις του, εκτός και αν προβλέπεται μηχανισμός exploration-exploitation.

2.4.1.1 Αλγόριθμος

Η Ενισχυτική Μάθηση (ΕΜ) όπως παρουσιάζεται στο διάγραμμα 2.2 έχει χαρακτηριστικά μιας πρωτόγονης μορφής μάθησης: Μέσω ανταμοιβής η τιμωρίας, ο πράκτορας έχει εικόνα αν αυτό που κάνει επιφέρει επιθυμητά αποτελέσματα μόνο μέσω της πληρωμής του. Πάντα προσαρμόζει τα βήματα του ώστε να έχει μέγιστη ανταμοιβή.



Διάγραμμα 2.2: Τυπικός βρόγχος ενισχυτικής μάθησης[1] - ο πράκτορας έχει ανάδραση από το περιβάλλον για την κατάσταση και την αμοιβή και αναλόγως προσαρμόζει τις επικείμενες ενέργειες του.

Τα βασικά στοιχεία της Ενισχυτικής Μάθησης είναι:

1. **Πράκτορας**

   Το ευφυές σύστημα που αφουγκράζεται το περιβάλλον του, σκέπτεται, προβλέπει, αυτοδιορθώνει παλαιότερες ενέργειες και έπειτα πράττει αντίστοιχα. Διαθέτει αισθητήρια για να μετρά κατάστασεις, λαμβάνει αμοιβή και εφαρμόζει ενέργειες προς το περιβάλλον του.

2. **Περιβάλλον**

   Ο χώρος μέσα στον οποίο εργάζεται ο Πράκτορας

3. **Πολιτική**

   Με την πολιτική ορίζουμε πως θέλουμε ο Πράκτορας να συμπεριφέρεται κάθε δεδομένη στιγμή - πρόκειται για μια αντιστοίχιση καταστάσεων του περιβάλλοντος με ενέργειες που πρέπει να κάνει.

4. **Συνάρτηση αμοιβής**

   Ορίζει τον στόχο σε ένα πρόβλημα Ενισχυτικής Μάθησης. Αντιστοιχεί ανταμοιβή ή τιμωρία σε μια ενέργεια και προσπαθεί έπειτα ο πράκτορας να μεγιστοποιήσει την συνολική θετική αμοιβή που θα δεχτεί.

5. **Συνάρτηση αξίας**

   Η αξία ή συνολική προβλεπόμενη στο μέλλον αμοιβή που θα λάβει ο πράκτορας μακροπρόθεσμα προέρχεται από την συνάρτηση αυτή. Η αμοιβή είναι άμμεση, η αξία είναι μακροπρόθεσμη.

6. **Μοντέλο**

   Το μοντέλο αναφέρεται στο περιβάλλον μέσα στο οποίο υπάρχει ο Πράκτορας. Μιμείται το περιβάλλον και χρησιμοποιείται για τον σχεδιασμό προβλέψεων μελλοντικών ενεργειών.

2.4.1.2 Q-Learning Πράκτορας

Ο Q-Learning είναι αλγόριθμος που χρησιμοποιεί τον παρακάτω κανόνα ενημέρωσης:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + a\left[r_{t+1} + \gamma \cdot \ max_a Q(s_{t+1}, a\ ) - Q(s_t, a_t)\right]$$

[1] Sutton Barto, "Reinforcement Learning: An Introduction", 1998

Ακολουθεί ο αλγόριθμος Q-Learning σε ψευδοκώδικα:

Initialize Q(s,a)
for each episode **do**
    Observe state s
    repeat
        Select action a evaluating Q
        Take action a
        Observe r, s'
        $Q(s,a) \leftarrow Q(s,a) + a\,[r + \gamma \cdot max_{a'} Q(s',a') - Q(s,a)]$
        s ← s'
    until s is terminal
end for

Κώδικας 2.1: ψευδοκώδικας που παρουσιάζει τον αλγόριθμο Q-Learning

2.4.1.3 Τυχαίος Πράκτορας

Ο Τυχαίος Πράκτορας[1] δοκιμάζει τυχαίες κινήσεις ώστε να εκπαιδεύσει το σύστημα για τυχόν περιπτώσεις που δεν έχουν καταγραφεί στο σύστημα από μια αρχική, τυποποιημένη εκπαίδευση.

2.4.1.4 Πράκτορας RLAgent του RLTankAttack

Ο Συνδυαστικός Πράκτορας του συστήματος RLTankAttack ακολουθεί αφαιρετικά την λογική της Ενισχυτικής Μάθησης. Αναζητά από μια αρχική λύση προς την νίκη βάση τυποποιημένης τυχαίας εκπαίδευσης. Εφόσον δεν βρεί προορισμό προς νίκη τότε και μόνο τότε επιλέγει τυχαία κίνηση μόνο εφόσον αυτή δοκιμαστεί και δεν επιφέρει στον Πράκτορα ήττα. Κατά τον τερματισμό του Πράκτορα, το παιχνίδι που ολοκληρώθηκε αναλύεται και ενημερώνεται ο γράφος πολιτικής με βάρη ανάλογα προς την ήττα, αντιστρόφως ανάλογα

[1] OSKAR ARVIDSSON, LINUS WALLGREN, "Q-Learning for a Simple Board Game", Sweden 2010 http://www.csc.kth.se/utbildning/kandidatexjobb/datateknik/2010/rapport/arvidsson_oskar_OCH_wallgren_linus_K10047.pdf

όταν καταλήγει σε νίκη. Αναλυτικότερα, ο αλγόριθμος του συστήματος κατά τον τερματισμό ενός παιχνιδιού έχει ως εξής:

```
LearnedFromGame(G): #G-->Game
    reward = GetReward(G)
    moves = LoadMovesDir(G)
    graph = LoadGraph( )
    weight = graph.MaxWeightGraph( )
    for move in moves:
        s_n, s_n_1 = state, state-1 #State at s_n and s_n-1
        a_n = action #Action leading from s_n-1 to s_n
        if reward == -1:
            weight = weight + 1
        if reward == 1:
            weight = weight - 1
        graph.AddEdge(s_n_1, s_n) = (a_n, weight)
```

Κώδικας 2.2: ψευδοκώδικας Python που παρουσιάζει τον αλγόριθμο RLAgent του RLTankAttack

2.4.2 Δομή Δεδομένων Πολιτικής Πράκτορα

Η πιο ευέλικτη δομή δεδομένων για χρήση αποτύπωσης καταστάσεων, ενεργειών και βαρών/ανταμοιβών είναι ο σταθμισμένος διγράφος (weighted directed graph). Πρόκειται για την δομή δεδομένων που μπορεί να αποθηκεύει την πολιτική του Πράκτορα.

2.4.2.1 Σταθμισμένος Διγράφος

Καταρχάς ένα παιχνίδι αποτυπώνεται με καταστάσεις, κίνηση και επόμενη κίνηση (ή αποτέλεσμα αυτής σε περίπτωση νίκης/ήττας). Μπορούμε να πούμε λοιπόν ότι η ροή κινήσεων είναι:

1. Κατάσταση παιχνιδιού κατά την κίνηση $t \rightarrow s_t$

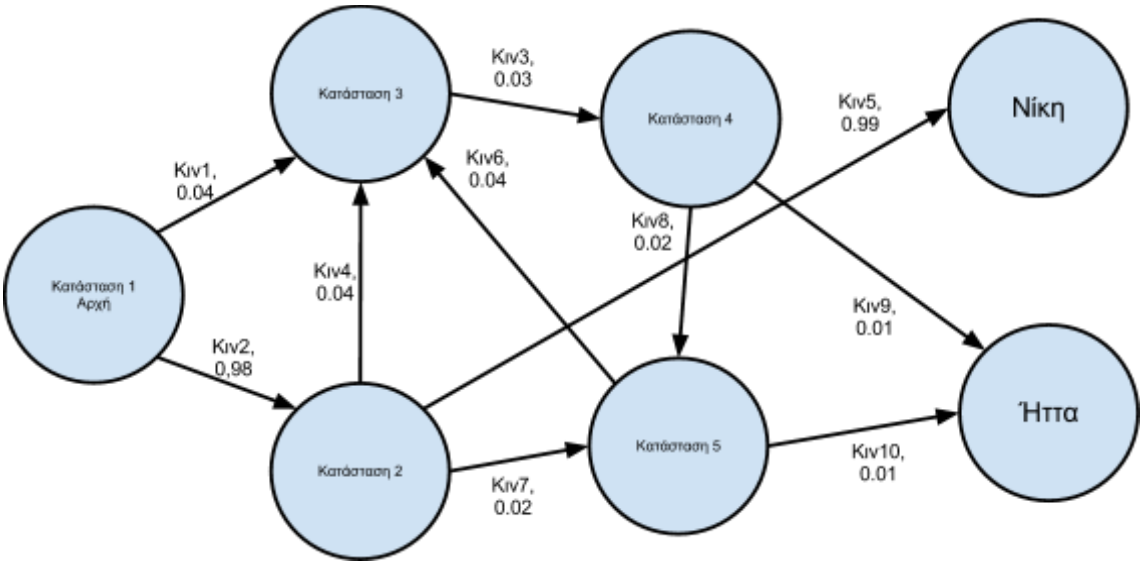
2. Κίνηση από κατάσταση t σε t+1 → a_t
3. Επόμενη κατάσταση → s_{t+1}
4. Τελική κατάσταση νίκης ή ήττας → s_{t+n}
5. Αμοιβή → r_{t+n}

Θεωρούμε λοιπόν σε ένα γράφο που θα κρατά πληροφορίες για το παιχνίδι ως $D = (V, A)$ με με κόμβους V και ακμές A που θα περιέχουν τις πληροφορίες:

V → s_t

A → (a_t, r_t)

Στο παρακάτω παράδειγμα παρουσιάζεται γράφος που αναπαριστά ένα παιχνίδι με 5 πιθανές καταστάσεις, 10 κινήσεις, 2 καταστάσεις τερματισμού.



Διάγραμμα 2.3: Σταθμισμένος διγράφος - Οι κόμβοι αποτυπώνουν καταστάσεις, οι ακμές περιέχουν την πληροφορία ότι η κίνηση με βάρος μας οδηγεί σε επικείμενη κατάσταση.

Ο παραπάνω γράφος είναι ουσιαστικά ο αρχικός χάρτης που χρησιμοποιεί ο Πράκτορας ώστε να γνωρίζει όλες τις πιθανές καταστάσεις και με ποια κίνηση/ενέργεια θα καταλήξει εκεί. Τα βάρη/αμοιβές ενισχύονται ή μειώνονται κατά την διάρκεια της μάθησης, καθώς θα ενημερώνει τον γράφο αυτό με κάθε ήττα ή νίκη. Κάθε κύκλος παιχνιδιού εμπλουτίζει τον γράφο και κάνει τον Πράκτορα ακόμα πιο προσαρμόσιμο.

2.5 Δοκιμές

Οι δοκιμές είναι το τρίτο και τελικό τμήμα της διατριβής καθώς έχει ως στόχο να αποτυπώσει τα αποτελέσματα των πειραμάτων και σεναρίων που θα προκύψουν από την δημιουργία του παιχνιδιού, του πράκτορα και του συνδυασμού τους. Ο πιο απλός τρόπος αποτύπωσης των δοκιμών είναι ο ορισμός δεικτών ποιότητας της συμπεριφοράς του Πράκτορα. Πρόκειται για μετρήσιμα μεγέθη που αντιστοιχούν και αποτυπώνουν εξέλιξη στην μάθηση του ευφυούς συστήματος.

2.5.1 Δείκτες Μάθησης

Μια απλή καταγραφή εκμάθησης ως προς το δημιουργηθέν σύστημα είναι η αντιπαραβολή έξυπνων (καταγεγραμμένων) κινήσεων έναντι των τυχαίων σε κάθε χρονική στιγμή, καθώς και η καταγραφή των ανταμοιβών σε βάθος χρόνου ανα παιχνίδι. Πρόκειται για τους πιο απλούς δείκτες ποιότητας του Πράκτορα:

2.5.1.1 Τύπος κίνησης

Σε κάθε παιχνίδι, κάθε κίνηση που πραγματοποιεί ο Πράκτορας μπορεί να είναι τυχαία (εφόσον δεν υπάρχει οδός προς την νίκη) ή ελεγχόμενη εφόσον αυτή είναι καταγεγραμμένη και υπάρχει προηγούμενο στον δίγραφο. Σε βάθος χρόνου θα πρέπει η πλειοψηφία των κινήσεων να είναι ελεγχόμενη.

2.5.1.2 Ανταμοιβή ανά παιχνίδι

Απλή αντιπαραβολή νίκης, ισοπαλίας ή ήττας σε κάθε παιχνίδι. Με το πέρας του χρόνου, θα πρέπει οι νίκες να εξισσοροπούνται.

Κεφάλαιο 3

Ανάπτυξη

Η ανάπτυξη βάσει σχεδίου πάντα προϋποθέτει να έχουν οριστεί τα απαραίτητα εργαλεία τα οποία θα βοηθήσουν στον σκοπό αυτό. Το πρώτο και σημαντικότερο εργαλείο, η βάση όλων σε ανάπτυξη σύγχρονων συστημάτων είναι η γλώσσα προγραμματισμού που θα χρησιμοποιηθεί. Θα πρέπει να είναι φιλική προς τον χρήστη, να είναι ευρείας αποδοχής ώστε να υπάρχουν τουλάχιστον οι κοινές και βασικές βιβλιοθήκες καθώς και να είναι γρήγορα επεκτάσιμη. Μια τέτοια σύγχρονη και ευέλικτη γλώσσα είναι η Python. Είναι η επιλογή πολλών εταιρειών αλλά και ερευνητικών έργων τα τελευταία χρόνια με μεγάλη επιτυχία.
Φυσικά τα παραπάνω ολοκληρώνονται με την επιλογή ενός ολοκληρωμένου περιβάλλοντος ανάπτυξης λογισμικού. Το πλεόν κατάλληλο είναι το Aptana Studio, το οποίο βασίζεται στο Eclipse της IBM.

3.1 Python και βιβλιοθήκες

3.1.1 Python

Η Python είναι μια γενικής χρήσης, υψηλού επιπέδου γλώσσα προγραμματισμού με χρήση μεταγλωττιστή. Έχει σχεδιαστεί ώστε ο κώδικας της να είναι εύκολα αναγνώσιμος και κατανοητός. Οι βιβλιοθήκες και η σύντομη περιγραφή τους αποτυπώνονται στον παρακάτω πίνακα:

Πρόγραμμα	**Περιγραφή**	**Παρατηρήσεις ανάπτυξης**
Python	Γλώσσα προγραμματισμού - βάση συστήματος	

twisted	Εργαλεία δικτυακής απλότητας	*προαιρετικό*
setuptools	Βασικό εργαλείο υποβοήθησης εγκατάστασης	
scipy	Επιστημονικά εργαλεία	
pywin32	Win32	*προαιρετικό*
pygame	Βιβλιοθήκη δημιουργίας παιχνιδιών	
numpy	Βελτιώσεις πινάκων	
networkx	Δημιουργία γράφων	
pydot	Δημιουργία γράφων σε dot	
pygraphviz	Γράφοι και παρουσίαση	
matplotlib	Παρουσίαση διαγραμμάτων	
pprint	Εκτυπωτής κονσόλας	
ttk	Themed Tkinter - Ενισχυμένο GUI για Python	

Πίνακας 3.1: Βιβλιοθήκες Python

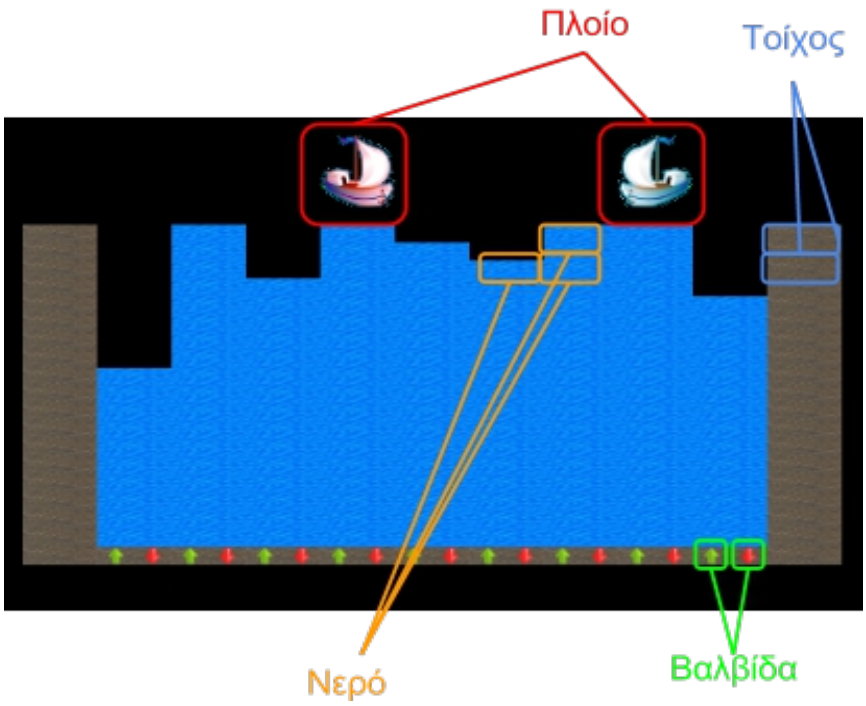
Η εγκατάσταση βιβλιοθηκών που δεν υπάρχουν διαθέσιμες σε *.exe ή repository πρέπει να εγκατασταθούν από τον πηγαίο κώδικα με μετάβαση στον κατάλογο της βιβλιοθήκης και εκτέλεση της εντολής:

```
python setup.py install
```

3.1.2 PyGame

Η βιβλιοθήκη PyGame δεν αποτελεί τμήμα της βασικής διανομής της Python. Η PyGame χρησιμοποιεί ως κύκλο ζωής ενός παιχνιδιού το διάγραμμα 2.1.

Κάθε αντικείμενο που θέλουμε να παρουσιάσουμε στην οθόνη μας μπορούμε να το ορίσουμε να είναι κλάσης Sprite, η οποία μας δίνει ιδιότητες διαστάσεων, σχήματος, χρώματος, κίνησης αλλά και ανανέωσης.
Πολλά Sprites στον καμβά μας οργανώνουν μια αποτύπωση του παιχνιδιού.



Εικόνα 3.1: Αποτύπωση αντικειμένων κλάσης Sprite του παιχνιδιού RLTankAttack

Στην παραπάνω εικόνα έχουμε την ταυτόχρονη χρήση μεταξύ των άλλων, πολλαπλών Sprites νερού, 2 πλοίου, τοίχου και βαλβίδας. Η παραπάνω οθόνη ανανεώνεται ~30 φορές το δευτερόλεπτο, και δεν ανανεώνει μέχρι να συμβεί κάποιο συμβάν όπως το πάτημα ενός κουμπιού ή κίνηση εισόδου κατάδειξης.

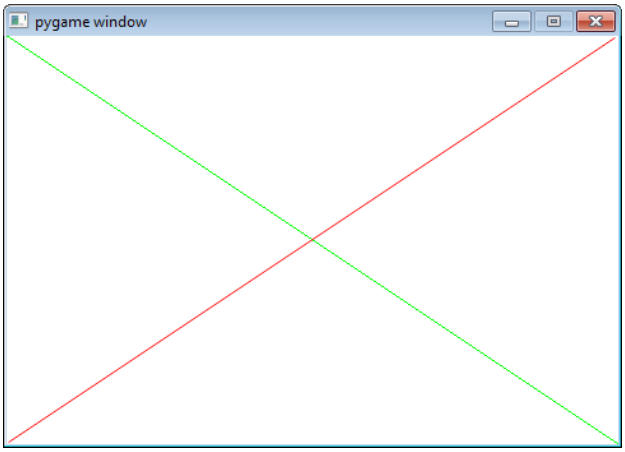
Ακολουθεί τμήμα κώδικα που παρουσιάζει την απλότητα μιας δημιουργίας εικόνας:

```
#! /usr/bin/env python
import pygame
screen = pygame.display.set_mode((480, 320))  #το παράθυρο μας
running = 1
while running:
   event = pygame.event.poll()  #αναμονή για συμβάν
   if event.type == pygame.QUIT:
      running = 0
```

```
screen.fill((255, 255, 255))  #γέμισμα παραθύρου με λευκό
pygame.draw.line(screen, (0, 255, 0), (0, 0), (479, 319))  #πράσινη γραμμή
pygame.draw.aaline(screen, (255, 0, 0), (479, 0), (0, 319))  #κόκκινη γραμμή
pygame.display.flip()  #ανανέωση οθόνης
```

Κώδικας 3.1: Πρόγραμμα Python & PyGame

Το παραπάνω πρόγραμμα έχει ως αποτέλεσμα την δημιουργία του ακόλουθου παραθύρου:



Εικόνα 3.2: Απλό πρόγραμμα PyGame

Το προτέρημα που μας δίνει η πλέον διαδεδομένη και εδραιωμένη βιβλιοθήκη παιχνιδιών PyGame είναι ότι χρησιμοποιεί κοινά πρότυπα της Python για γρήγορη ανάπτυξη χωρίς να απασχολεί τον προγραμματιστή σε χρονοβόρες λεπτομέρειες όπως η ενημέρωση οθόνης, τον διακόπτη συμβάντων κλπ.

3.1.3NumPy

Η βιβλιοθήκη NumPy (Numerical Python) μας δίνει μια πληθώρα υπολογιστικών εργαλείων για την Python. Μεταξύ των άλλων περιέχει:

1. Πίνακας-Αντικείμενο N-διαστάσεων
1. Συναρτήσεις διαχείρισης, αναζήτησης
2. Διεπαφή με άλλες γλώσσες προγραμματισμού
3. Λειτουργίες γραμμικής άλγεβρας, μετατροπών Fourier και συναρτήσεων και γεννητριών τυχαίων αριθμών

Ένα παράδειγμα ευκολίας της NumPy είναι η αναζήτηση τιμής μέσω της συνάρτησης where:

```
numpy.where(condition[, x, y]) #returns ndarray or tuple of ndarrays
```

Η ευκολία που προσφέρει είναι προφανής, καθώς η αναζήτηση όρου είναι τάχιστη και αποτελείται μόνο από μια γραμμή κώδικα.

3.1.4 NetworkX

Η βιβλιοθήκη NetworkX, είναι μια ολοκληρωμένη προσπάθεια του Εθνικού Εργαστηρίου LANL για Python που βοηθά στην δημιουργία, χειρισμό και μελέτη δομών, δυναμικής και λειτουργιών πολύπλοκων δικτύων και γράφων. Η απλότητα της και η χρήση της αποτυπώνεται στο τμήμα Κώδικα 3.2:

```
#! /usr/bin/env python
import networkx as nx
G=nx.Graph()
G.add_edge(1,2)
```

Κώδικας 3.2: Πρόγραμμα Python & Networkx

3.1.5 PyDot

Η γλώσσα Dot είναι μια περιγραφική γλώσσα κειμένου για την περιγραφή γράφων. Πρόκειται δηλαδή για τον χάρτη ενός γράφου, των κόμβων, ακμών και ιδιοτήτων του για χρήση από άλλα προγράμματα. Είναι δηλαδή ένα πρότυπο ορισμού γράφων.

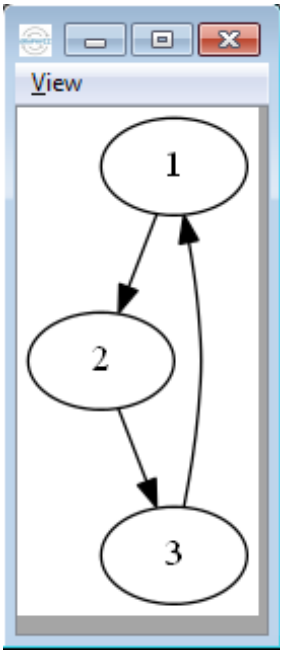
```
digraph G {
 1->2;
 2->3;
 3->1;
}
```

Κώδικας 3.3: Κώδικας Dot

Η βιβλιοθήκη PyDot μεταφέρει την επικοινωνία από και προς DOT σύνταξη σε Python.

3.1.6 GraphViz & PyGraphViz

Ένα ώριμο λογισμικό οπτικοποίησης γράφων είναι το GraphViz. Το απόσπασμα κώδικα 3.3 οπτικοποιείται:



Εικόνα 3.3: Οπτικοποίηση Κώδικα 3.3 μέσω GVEdit για GraphViz

Το Εθνικό Εργαστήριο LANL διαθέτει διεπαφή ονόματι PyGraphViz που ενσωματώνει τις δυνατότητες του εργαλείου αυτού στο σύστημα μας.

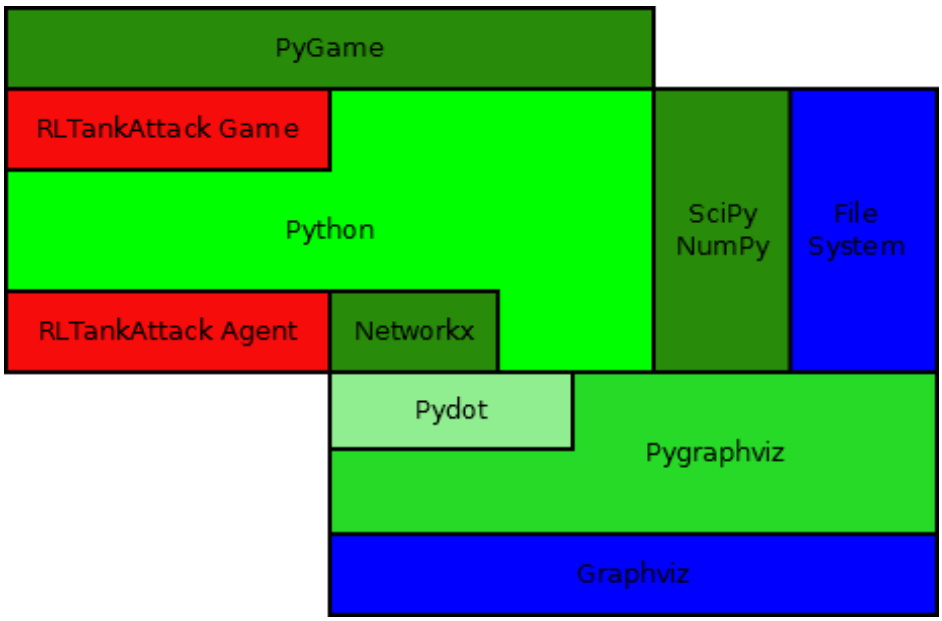
Σημείωση: Η εγκατάσταση του PyGraphViz σε Λειτουργικό Σύστημα Windows Xp/7 είναι αρκετά πολύπλοκη, για αυτό το σύστημα διαθέτει Plotter γράφων με χρήση τόσο του PyGraphViz όσο και του MatPlotLib. Περισσότερα στο http://networkx.lanl.gov/pygraphviz/

3.1.7 MatPlotLib

Άλλο ένα λογισμικό οπτικοποίησης της Python. Έχει την δυνατότητα εκτός από γράφους να δημιουργεί διαγράμματα, γραφικές παραστάσεις και απεικονίσεις όπως η εικόνα 2.3. Είναι η πιο διαδεδομένη και ευρύας γκάμας βιβλιοθήκη αναπαράστασης γραφημάτων, δικτύων κλπ. Περισσότερα για την βιβλιοθήκη: http://matplotlib.org/

3.1.8 Αρχιτεκτονική Σχεδίαση RLTankAttack

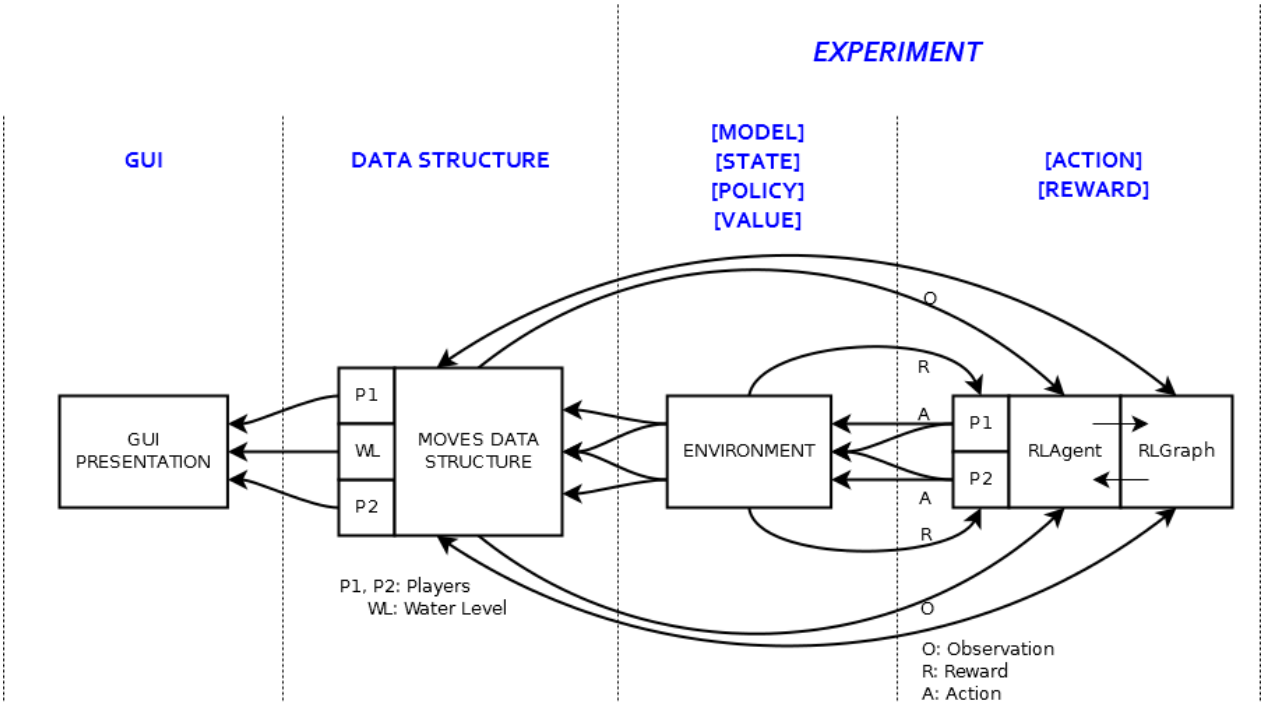
Όλα τα παραπάνω έρχονται και δένουν στην ακόλουθη στοίβα ενοποίησης:



Εικόνα 3.4: Αφαιρετική στοίβα Python, Βιβλιοθηκών ως προς το RLTankAttack

Με κόκκινο γέμισμα είναι ορατό το παιχνίδι και ο πράκτορας. Η επικοινωνία τους γίνεται μέσω Python και των αρχείων καταγραφής (§2.3.6).

Η ροή επικοινωνίας αποτυπώνεται στο ακόλουθο διάγραμμα:



Διάγραμμα 3.1: Αφαιρετική στοίβα επικοινωνίας Παιχνιδιού και Πράκτορα

Όπως φαίνεται στο διάγραμμα 3.1, το παιχνίδι με τον έλεγχο των αρχείων καταγραφής βρίσκονται στις 2 αριστερές στήλες GUI και DATA STRUCTURE. Οι δύο δεξιές στήλες

περιέχουν τις λειτουργίες του πράκτορα με την δημιουργία και διαχείριση του γράφου του παιχνιδιού και την αλληλεπίδραση με τα αρχεία καταγραφής.

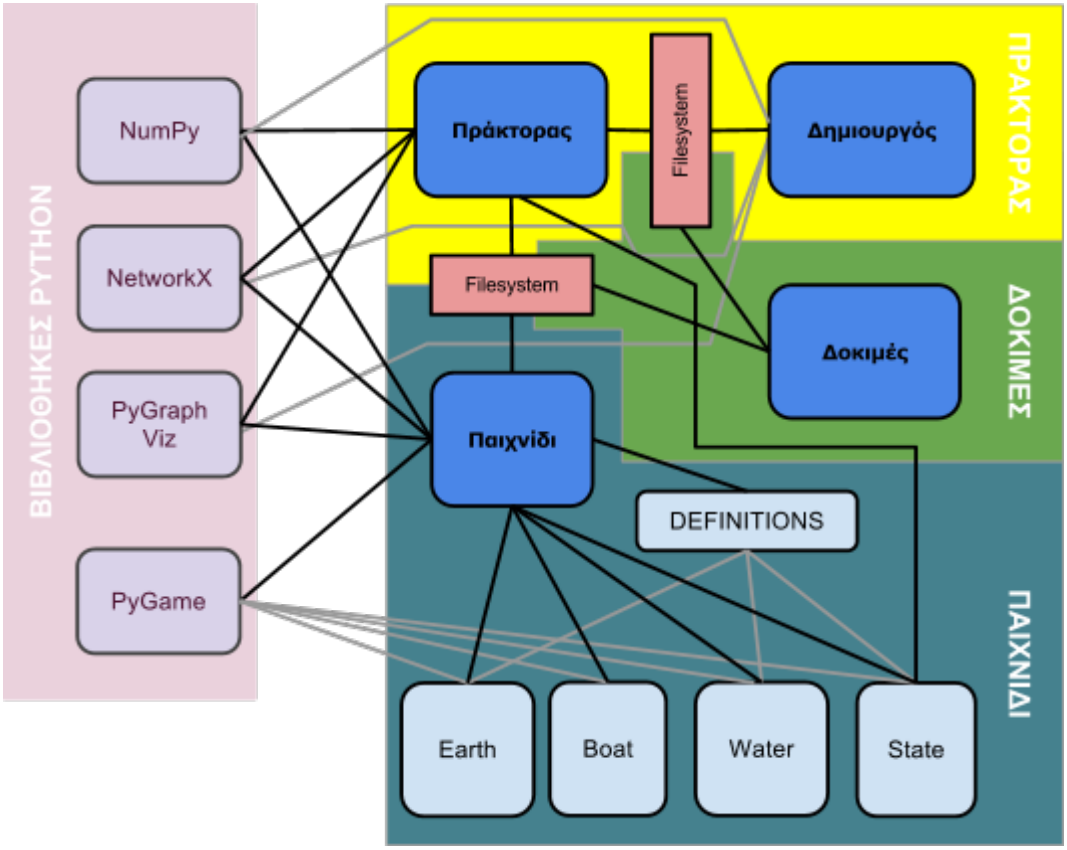
3.2 Παιχνίδι

Το παιχνίδι είναι η διεπαφή όπου είναι τόσο οπτικά όσο και αλγοριθμικά η αλληλεπίδραση παικτών με βάση κάποιο στόχο. Ο παίκτης μπορεί να είναι άνθρωπος ή πράκτορας.

3.2.1 Οντότητες παιχνιδιού

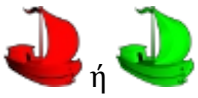
Οι βασικές λογικές οντότητες έχουν κωδικοποιούνται στο διάγραμμα 3.1 που ακολουθεί ως εξής:

ΜΠΛΕ ΦΟΝΤΟ: Το παιχνίδι και μερικές από τις υλοποιημένες
ΚΙΤΡΙΝΟ ΦΟΝΤΟ: Πράκτορας και υλοποιήσεις δημιουργίας Περιβάλλοντος.
ΠΡΑΣΙΝΟ ΦΟΝΤΟ: Οι δοκιμές που γίνονται από τα αρχεία καταγραφής και τον γράφο
ΡΟΖ ΦΟΝΤΟ: Η Python και οι τυποποιημένες βιβλιοθήκες



Διάγραμμα 3.2: Υλοποιημένες Οντότητες - Παιχνίδι, Πράκτορας & Δοκιμές

Το παιχνίδι παρουσιάζεται στο μπλέ φόντο και μεταξύ των άλλων περιέχει επιγραμματικά τις παρακάτω κύριες κλάσεις αντικειμένων:

Κλάση	Υπερκλάση	UI Εικονίδιο	Μέθοδοι
Βάρκα *Boat.py*	pygame.sprite.Sprite	 ή	1. Κίνηση 1. Δεξιά 2. Αριστερά 3. Πάνω 4. Κάτω 2. Σχεδίαση 3. Θέση
Νερό *Water.py*	pygame.sprite.Sprite		1. Σχεδίαση 2. Θέση
Στήλη Νερού *WaterLevel.py*	-	-	1. Κίνηση 1. Ώθηση νερού 2. Απώθηση νερού
Βαλβίδα *Valve.py*	pygame.sprite.Sprite	 ή	1. Σχεδίαση 2. Θέση

			1. Στήλη
Κατάσταση Παιχνιδιού *GameState.py*	-	-	1. Αμοιβή για κίνηση 2. Κατάσταση
Σειρά παίκτη WhosTurn.py	pygame.sprite.Sprite	 ή	1. Εναλλαγή σειράς 2. σχεδίαση
Παιχνίδι *RunGame.py*	-	-	1. Εκκίνηση κύκλου παιχνιδιού PyGame όπως στο διάγραμμα 2.1

Πίνακας 3.2: Κύριες κλάσεις αντικειμένων RLTankAttack

3.2.2 Κλάσσεις Λογικής

3.2.2.1 Κανόνες

Οι κανόνες περιέχονται στο GameState.py όπου γίνεται έλεγχος του αποτελέσματος της κίνησης (η κίνηση ελέγχεται για νομιμότητα επιλογής αντικειμένου, πλήκτρων κλπ από το ίδιο το παιχνίδι).

Διάγραμμα 3.3: Κύκλος ελέγχου κατάστασης μετά από κίνηση

Μέσα από τους κανόνες του πιος και πότε είναι νικητής, η κλάση αυτή ελέγχει και επιστρέφει κατάσταση παιχνιδιού και πλασματική αμοιβή ή αποτέλεσμα με τον ακόλουθο κανόνα:

Τιμή **Περιγραφή**

-1 → Ήττα Πράκτορα

0 → Δεν υπάρχει ακόμη νικητής - το παιχνίδι δεν έχει τερματίσει και συνεχίζεται

1 → Νίκη Πράκτορα

3.2.3 Κύκλος ζωής

Ο κύκλος ζωής του παιχνιδιού αναφέρεται επιγραμματικά στο §2.3.1. Από τα πιο σημαντικά είναι τα συμβάντα και η νομιμότητα αυτού.

3.2.3.1 Συμβάντα παιχνιδιού

Κληρονομημένο από από την PyGame, το Παιχνίδι (RunGame.py) αναμένει συμβάν από το σύστημα.

Αποδεκτά συμβάντα

1. Κλικ από ποντίκι
    1. Πάνω σε Πλοίο
    2. Πάνω σε Νερό
    3. Πάνω στο κουμπί τερματισμού του παραθύρου

Όλα τα άλλα συμβάντα εξαιρούνται και δεν γίνεται επεξεργασία τους και απορρίπτονται στον περαιτέρω έλεγχο για το παιχνίδι μας.

3.3 Πράκτορας

Ο οντότητα ΠΡΑΚΤΟΡΑΣ όπως φαίνεται στο διάγραμμα 3.1 με κίτρινο φόντο αποτελείται από δύο τμήματα: Τον "*Δημιουργό*" και τον "*Παίκτη*". ο Δημιουργός οικοδομεί την αρχική αποτύπωση του περιβάλλοντος σε μορφή σταθμισμένου διγράφου. Ο Πράκτορας είναι ο παίκτης που παίζει ως αντίπαλος με με το πλοίο #1 (#0 είναι πάντα άνθρωπος). Ο Πράκτορας προαπαιτεί να έχει γίνει εργασία από τον Δημιουργό ώστε να γνωρίζει το περιβάλλον στο οποίο έχει να παίξει.

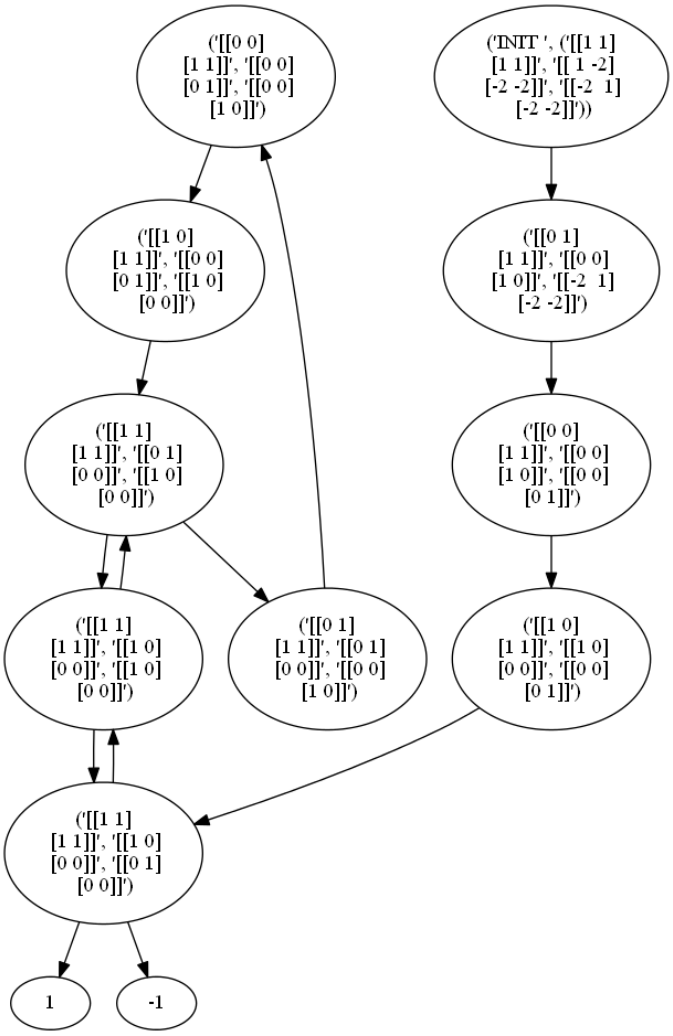
3.3.1 Δημιουργός Αρχικής Πολιτικής

Ο Δημιουργός αρχικής Πολιτικής κατασκευάζει ένα σταθμισμένο διγράφο που περιέχει κινήσεις-ενέργειες-αμοιβές. Υλοποιείται από το και στην πράξη υλοποιεί ότι αναφέρεται στην Σχεδίαση §2.4.2.

Ακολουθούνται δυο μέθοδοι δημιουργίας τέτοιου γράφου: Επαναληπτικού Βρόγχου και Γεννήτρια Τυχαίων Κινήσεων.

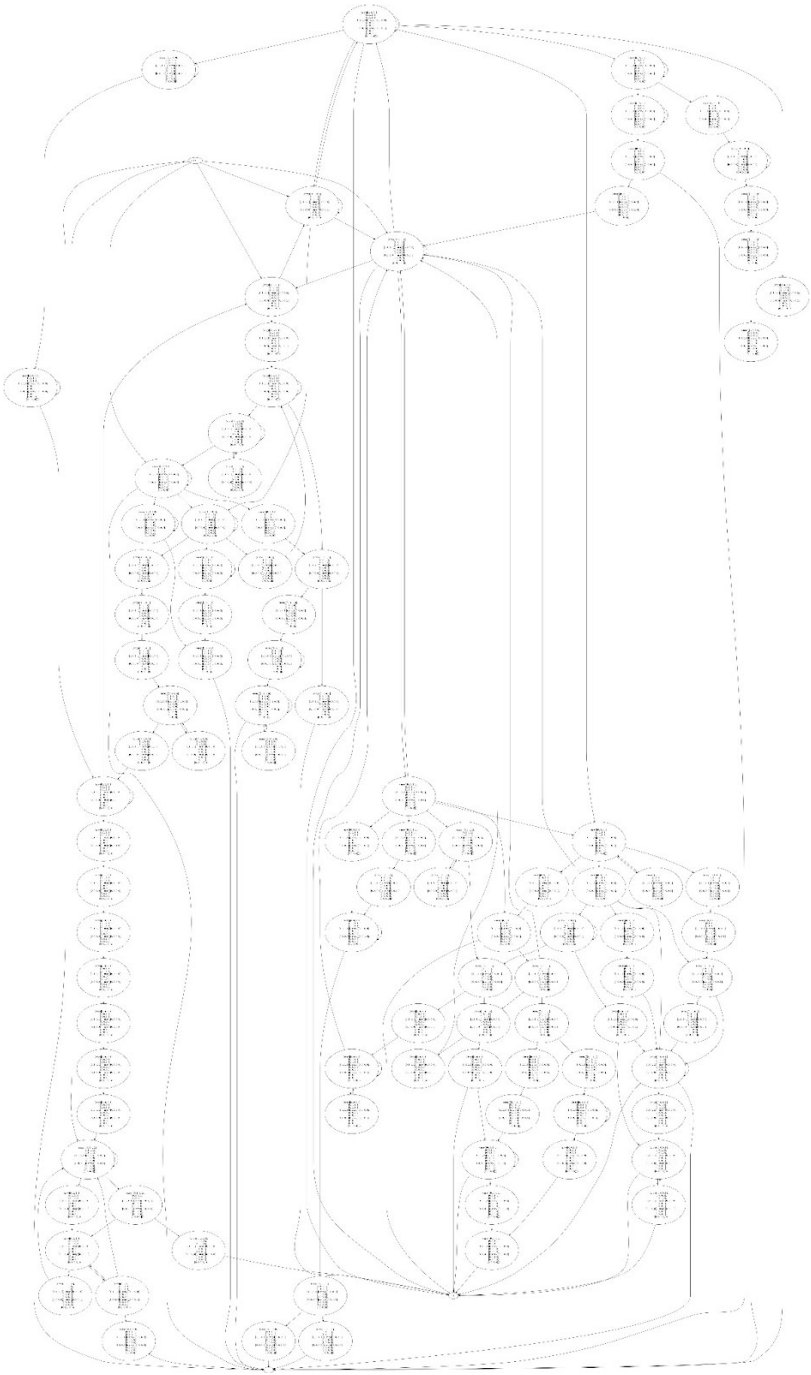
3.3.1.1 Δημιουργός Επαναληπτικού Βρόγχου

Ακολουθούν οπτικοποιημένοι γράφοι που έχουν κατασκευαστεί για παιχνίδι 2x2 και 4x4:



Διάγραμμα 3.4: Γράφος παιχνιδιού 2x2 - απεικονίζει καταστάσεις B^0_{2x2}, B^1_{2x2} & WL_{2x2} - δεν απεικονίζονται οι ακμές με τις ενέργειες και τα βάρη. Δημιουργείται από το πρόγραμμα RLGraphBuilder.py.

Αντίστοιχη αλλά ποιο πολύπλοκη είναι η εικόνα του περιβάλλοντος μας για παιχνίδι 4x4:



Διάγραμμα 3.5: Γράφος παιχνιδιού 4x4 - απεικονίζει καταστάσεις B^0_{4x4}, B^1_{4x4} & WL_{4x4} - δεν απεικονίζονται οι ακμές με τις ενέργειες και τα βάρη.

Η πολυπλοκότητα και εξέλιξη των κόμβων θα αναλυθεί στην παράγραφο 3.4.

Ο συνολικός αριθμός των κινήσεων είναι:

$2 \cdot [?][?]^{[?][?]} - 2$, Μ στήλες, Ν γραμμές

3.3.1.2 Δημιουργός Γεννήτριας Τυχαίων Κινήσεων

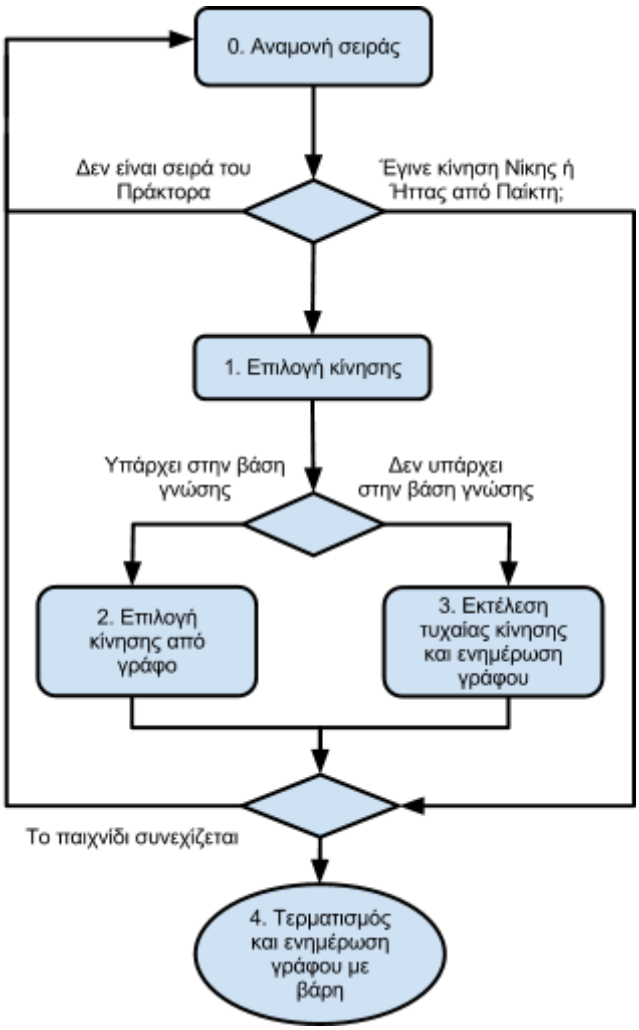
Εκτός του κλασσικού επαναληπτικού βρόγχου προσθήκης κινήσεων, κατασκευάστηκε και Δημιουργός Αρχικής Πολιτικής με γεννήτρια Χ Τυχαίων Κινήσεων. Σε αυτή την περίπτωση αποστέλλονται προς δοκιμή νομιμότητας τυχαίες κινήσεις συνεχόμενα για 10000 επαναλήψεις. Αναλυτικότερα τα αποτελέσματα επιλογής των επαναλήψεων ΄βρίσκονται στην παράγραφο 3.4.2.2 Ενέργειες Πράκτορα.

3.3.2 Πράκτορας Παίκτης

Ο Πράκτορας Παίκτης που έχει δημιουργηθεί είναι Συνδυαστικός Πράκτορας Ενισχυτικής Μάθησης και Επιλογής Τυχαίας Κίνησης. Στις κινήσεις που αποφασίζει να κάνει βάσει γνώσης κινείται επιθετικά, στις κινήσεις που δεν γνωρίζει δρόμο προς νίκη τηρεί αμυντική στάση με επιλογή τυχαίας κίνησης που δεν θα του επιφέρει άμεση ήττα.

3.3.2.1 Κύκλος ζωής

Ο Πράκτορας αναμένει κινήσεις βάσει των αρχείων καταγραφής. Αναμένει την σειρά του και εφόσον είναι δική του σειρά τότε βάσει του γράφου γίνεται εκτέλεση ενέργειας. Αν και πάλι δεν υπάρχει μια διαθέσιμη κίνηση που να οδηγεί σε νίκη από την παρούσα κατάσταση, τότε εκτελείται τυχαία κίνηση και καταγράφεται στον διγράφο. Τέλος, μετά τον τερματισμό του παιχνιδιού, ο Πράκτορας ενημερώνει για τις τελευταίες κινήσεις του και το που τον κατέληξαν αυτές τα βάρη του γράφου μας ώστε σε περίπτωση ήττας να έχουμε πολύ μικρή πιθανότητα επιλογής και για νίκη μεγαλύτερη για επόμενα παιχνίδια.



Διάγραμμα 3.6: Λογικό διάγραμμα ενημέρωσης γράφου με τυχαίες κινήσεις καθώς και με βάρη κατά τον τερματισμό.

Για το παραπάνω βήμα *"4. Εκτέλεση τυχαίας κίνησης"* ακολουθείται ο παρακάτω αλγόριθμος επιλογής τυχαίας κίνησης:

```
ΕκτέλεσηΤυχαίαςΚίνησης(Παιχνίδι, Κίνηση):
    Επιλογή Τυχαία_Κίνηση
    Αμοιβή = Κατάσταση(Τυχαία_Κίνηση)
    Αμοιβή ==-1:
        ΕκτέλεσηΤυχαίαςΚίνησης(Παιχνίδι, Κίνηση)
    Αμοιβή >=0:
        ΕπιστροφήΚίνησης(Τυχαία_Κίνηση)
```

Κώδικας 3.4: Ψευδοκώδικας Επιλογής Τυχαίας Κίνησης

3.4 Δοκιμές

Οι δοκιμές αποτυπώνουν πειράματα, δεδομένα αλλά και συμπεράσματα για την ανάπτυξη και συμπεριφορά του Παιχνιδιού αλλά και του Πράκτορα.

3.4.1 Καταγραφή παιχνιδιού

3.4.1.1 Επικοινωνία Πράκτορα

Τόσο η νίκη του πράκτορα όσο και η ήττα καταγράφονται και ενημερώνεται στα αρχεία καταγραφής. Η όλη επικοινωνία γίνεται αποκλειστικά από αυτά τα αρχεία.\

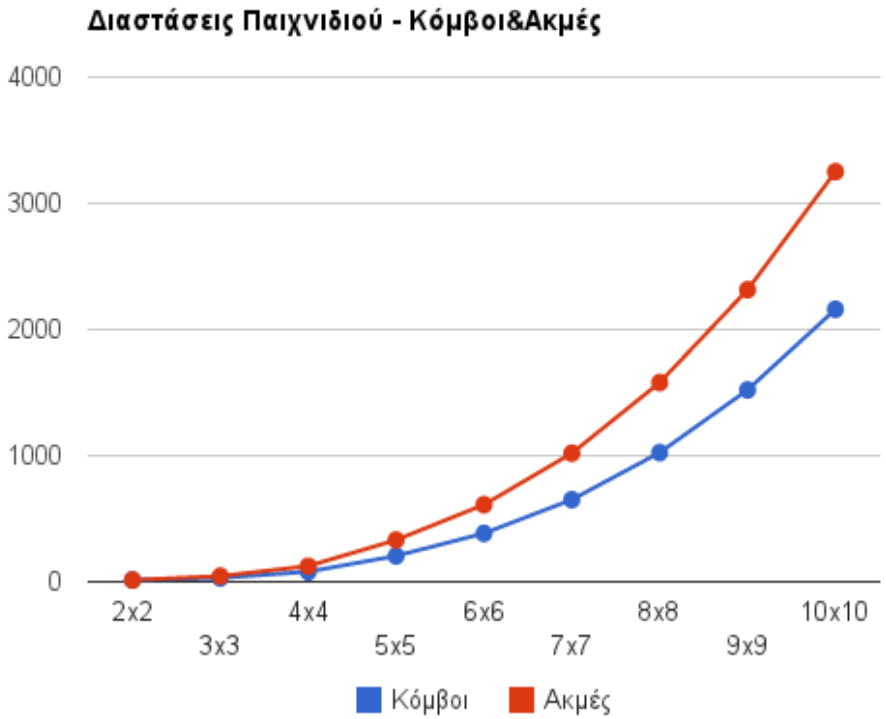
3.4.2.2 Ενέργειες Πράκτορα

Οι διαθέσιμες ενέργειες που μπορεί να κάνει ο Πράκτορας μας αναφέρεται στις ακμές του γράφου περιβάλλοντος. Μετά από καταγραφή των στοιχείων ΚΟΜΒΟΙ & ΑΚΜΕΣ του γράφου που κατασκευάζει ο Δημιουργός, τα αποτελέσματα είναι:

Μέγεθος	Κόμβοι	Ακμές
2x2	14	14
3x3	30	45
4x4	80	125
5x5	205	331
6x6	384	611
7x7	651	1019
8x8	1024	1579
9x9	1521	2315
10x10	2160	3251

Πίνακας 3.3: Αριθμός κόμβων και ακμών γράφου ανάλογα με τις διαστάσεις MxN των στοιχείων νερού του παιχνιδιού.

Τα παραπάνω οπτικοποιούνται στο παρακάτω γράφημα:



Γράφημα 3.1: Αριθμός κόμβων και ακμών γράφου περιβάλλοντος ανάλογα με τις διαστάσεις του παιχνιδιού.

Η πολυπλοκότητα αυξάνεται εκθετικά όπως είναι εμφανές από το γράφημα 3.1: όσο μεγαλύτερες οι διαστάσεις του παιχνιδιού, τόσο πιο δύσκολη η παραγωγή γράφου αλλά και ο εμπλουτισμός με γνώση και βάρη πάνω στο περιβάλλον μας.

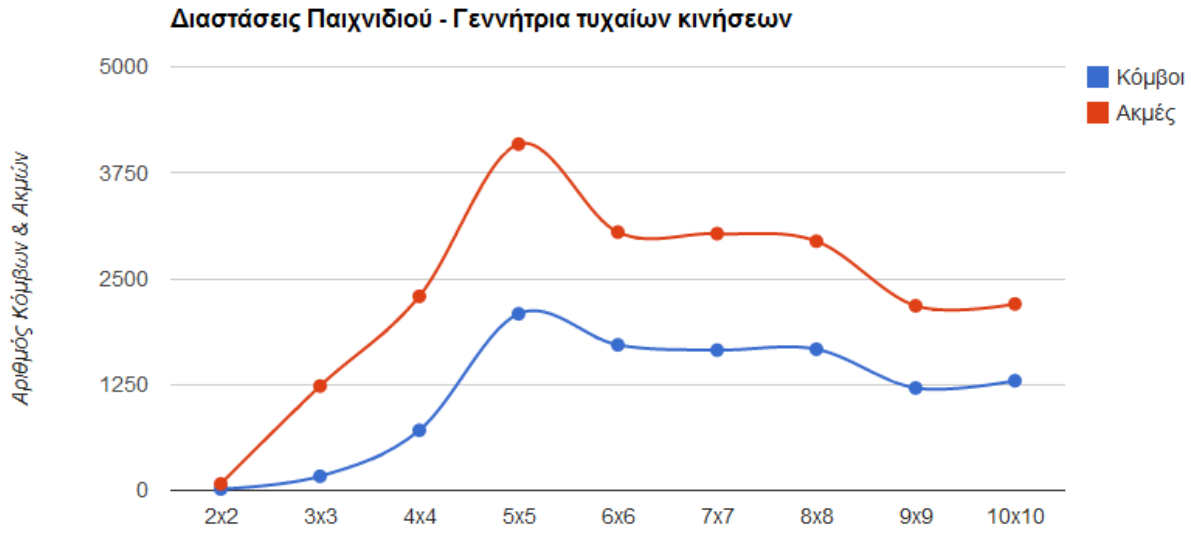
Στην περίπτωση που ο Δημιουργός Πολιτικής είναι γεννήτρια τυχαίων κινήσεων έχουμε ακόμη μεγαλύτερο και πληρέστερο γράφο. Στη περίπτωση σάρωσης 10000 τυχαίων κινήσεων και καταγραφής αυτών έχουμε:

Μέγεθος	**Κόμβοι**	**Ακμές**
2x2	16	77
3x3	167	1232
4x4	709	2291
5x5	2085	4088
6x6	1718	3049
7x7	1656	3031
8x8	1667	2942

9x9	1209	2179
10x10	1295	2201

Πίνακας 3.4: Αριθμός κόμβων και ακμών γράφου ανάλογα με τις διαστάσεις MxN των στοιχείων νερού του παιχνιδιού.

Τα παραπάνω οπτικοποιούνται στο παρακάτω γράφημα:



Γράφημα 3.2: Αριθμός κόμβων και ακμών γράφου περιβάλλοντος ανάλογα με τις διαστάσεις του παιχνιδιού βάσει 10000 τυχαίων κινήσεων.
Η πολυπλοκότητα αυξάνεται αλλά απαιτεί περισσότερα τυχαία περάσματα όσο μεγαλώνουν οι διαστάσεις του παιχνιδιού.

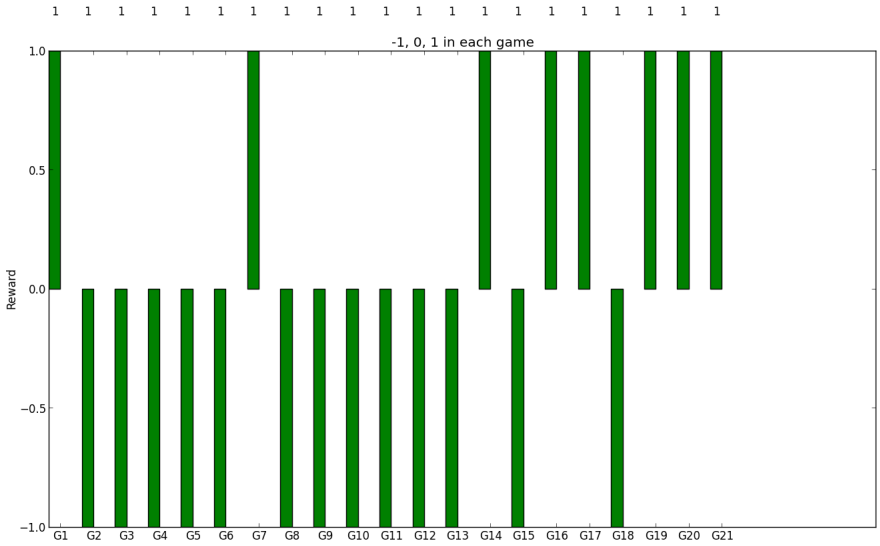
3.4.2.3 Δείκτες Μάθησης

Ο ρυθμός μάθησης παρακολουθείται με επιτυχία με τα ακόλουθα βήματα:

1. Καταγραφή κινήσεων
2. Κατηγοριοποίηση κίνησης
3. Καταγραφή αποτελέσματος κίνησης

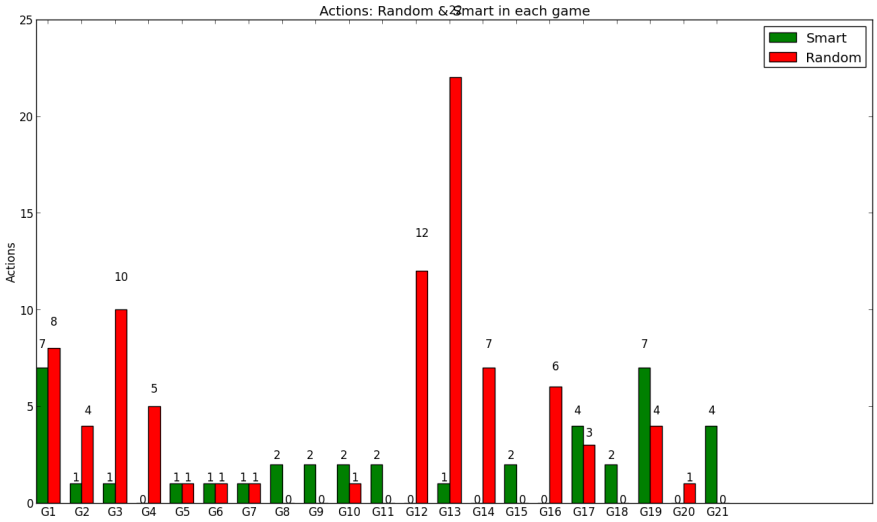
Σε βάθος 100-1000 παιχνιδιών είναι εφικτό να προκύψουν ικανοποιητικά συμπεράσματα για το αν το σύστημα επίλυσης προβλημάτων έχει υλοποιηθεί σωστά. Αρχικοί δείκτες που πρέπει να καταγράφονται αναφέρονται στην παράγραφο 2.5.1 Δείκτες Ποιότητας.

Μια απεικόνηση αντιπαραβολής αμοιβών ακολουθεί:



Εικόνα 3.5: Εξέλιξη δείκτη αμοιβών Πράκτορα

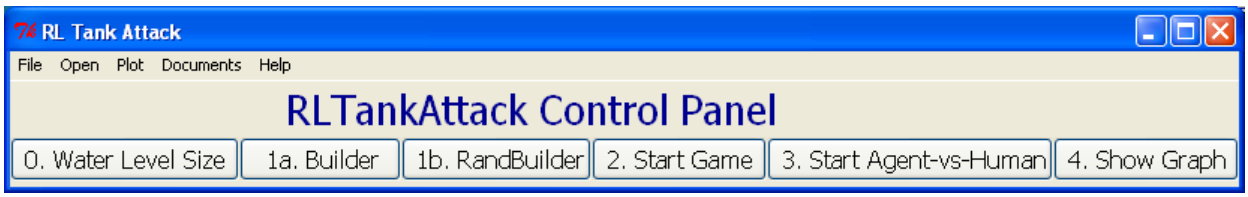
Επίσης ακολουθεί στην εικόνα 3.6 ο δείκτης τύπου επιλογών του Πράκτορα:



Εικόνα 3.6: Εξέλιξη δείκτη τύπου ενεργειών Πράκτορα

3.4.3 Παρουσίαση αποτελεσμάτων - Control Panel

Για την καλύτερη παρουσίαση του συστήματος RLTankAttack, αναπτύχθηκε βασικό GUI το οποίο συγκεντρώνει τα βήματα παρουσίασης ενός παιχνιδιού με αντίπαλο και παρουσίασης εξέλιξης συλλογής γνώσης:



Εικόνα 3.7: Control Panel για το σύστημα RLTankAttack

Τα βήματα εκτέλεσης του συστήματος είναι:

0. Water Level Size

Διαστάσεις μεγέθους πίνακα δεξαμενής νερού

1a. PolicyBuilder

Εκτέλεση Δημιουργού Πολιτικής - κατασκευή αρχικού γράφου καταστάσεων-ενεργειών

1b. RandPolicyBuilder

Εκτέλεση Δημιουργού Πολιτικής - κατασκευή αρχικού γράφου καταστάσεων-ενεργειών με γεννήτρια τυχαίων κινήσεων

2. Start Game

Εκτέλεση παιχνιδιού με διαστάσεις οριζόμενες από το 0

3. Start Agent-vs-Human

Εκκίνηση Πράκτορα - προϋποθέτει το βήμα 2 να είναι σε εκτέλεση

4. Show Graph

Παρουσίαση ανα πάσα στιγμή του γράφου πολιτικής - προϋποθέτει την εκτέλεση τουλάχιστον μιας φοράς του βήματος 1.

Τα στοιχεία του μενού επιλογής Open μας οδηγούν στο άνοιγμα των καταλόγων:

1. Γράφου
2. Αρχείων dot
3. Αποτυπώσεως png των αντίστοιχων dots αρχείων
4. Κινήσεων παιχνιδιών σε σειριακά αρχεία

Κεφάλαιο 4

Αποσφαλμάτωση

4.1 Διαδικασία αποσφαλμάτωσης

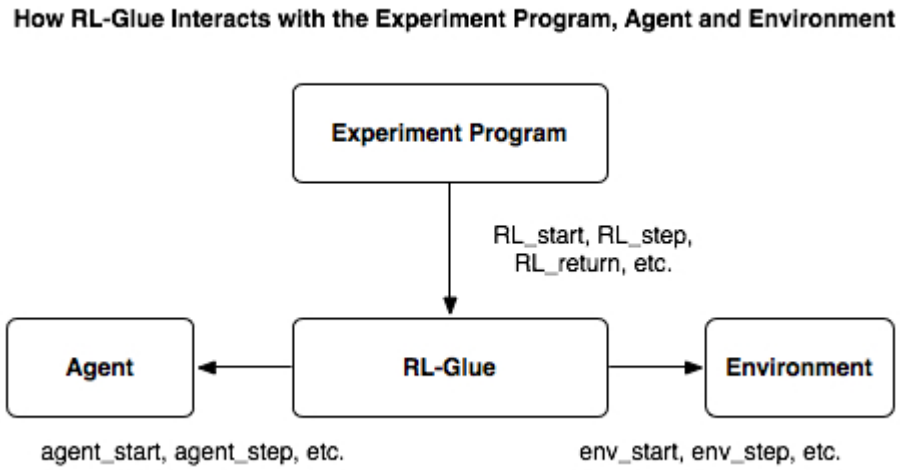
Η διαδικασία αποσφαλμάτωσης του όλου συστήματος απαιτεί συνεχή καταγραφή των ενεργειών που γίνονται σε κάθε φάση και σε κάθε τομέα ξεχωριστά. Οι δυο διακριτές φάσεις είναι η Σχεδίαση και Ανάπτυξη. Οι τομείς είναι το Παιχνίδι και ο Πράκτορας.
Σε κάθε κομμάτι λοιπόν απαιτείται ενδελεχής αποτύπωση των εργασιών για την ανίχνευση, διάγνωση και επίλυση επιμέρους προβλημάτων. Εδώ ανακύπτει και το βασικό πρόβλημα της Python: Κώδικας ο οποίος δεν εκτελείται δεν δοκιμάζεται! Τι σημαίνει αυτό; Σε περίπτωση που μια εντολή μας βρίσκεται σε μια επιλογή που για τον Α ή Β λόγο δεν μας κατέληξε το παιχνίδι ή ο πράκτορας σε εκείνη την πιθανότητα (if statement) τότε η Python αγνοεί την ύπαρξη των εντολών.
Υπάρχουν 2 μέθοδοι για την αντιμετώπιση αυτής της ατέλειας (ευλογίας κατά πολλούς): Ενδελεχής έλεγχος κώδικα και διεξοδική εκτέλεση πιθανών εντολών ή χρήση πακέτου Δοκιμών Μονάδων (Unit Testing). Ένα τέτοιο απλό άρθρωμα είναι το unittest (PyUnit). Αρκετές συναρτήσεις του συστήματος RLTankAttack δοκιμάστηκαν, παρόλα αυτά αποτελούν χρονοβόρο και επίπονο κομμάτι της ανάπτυξης λογισμικού. Για αυτό τον λόγο, η πλειοψηφία του συστήματος διαθέτει ιδιόχειρα συστήματα αποσφαλμάτωσης.

4.2 Εναλλακτικές υλοποιήσεις και λόγοι απόρριψης

Οι κύριες πλατφόρμες που υπάρχουν αυτή την στιγμή και υλοποιούν ΄συστηματικά Ενισχυτική Μάθηση (μεταξύ άλλων μεθόδων) είναι το RL-Glue (αποκλειστικό συνδετική λογισμική στοίβα Ε.Μ) και το PyBrain.

4.2.1 RL-Glue

Το RL-Glue[1] αποτελεί ένα ανεξάρτητο, στάνταρ διεπαφής σύνδεσης Ενισχυτικής Μάθησης στα τρεις χαρακτηριστικές οντότητες Πράκτορας-Περιβάλλον-Πείραμα. Στα πλεονεκτήματα του συστήματος είναι ότι ασχολείται αποκλειστικά με Ενισχυτική Μάθηση, διαθέτει διεπαφές σε αρκετές γλώσσες προγραμματισμού και αποτελείται από μεγάλη επιστημονική και φοιτητική κοινότητα.



Εικόνα 4.1: Διεπαφή Πειράματος, Πράκτορα και Περιβάλλοντος

4.2.2 PyBrain

Το PyBrain[2] είναι μια ευρύτερη βιβλιοθήκη που εμπεριέχει διάφορες μεθόδους Μηχανικής Μάθησης όπως είναι Η Ενισχυτική, Νευρωνικών Δικτύων κ.α.

4.2.3 Λόγοι Απόρριψης

Κύριος λόγος απόρριψης ήταν η αρκετά μεγάλος χρόνος που απαιτούσαν στην μάθηση τους (learning curve) αλλά και ενσωμάτωση τους στο υπάρχων σύστημα. Σε ανάπτυξη

[1] Brian Tanner and Adam White. RL-Glue: Language-Independent Software for Reinforcement-Learning Experiments. Journal of Machine Learning Research, 10(Sep):2133--2136, 2009. (BibTex)

[2] Schaul, Tom & Bayer, Justin & Wierstra, Daan & Sun, Yi & Felder, Martin & Sehnke, Frank & Ruckstiess, Thomas & Schmidhuber, Jurgen, "Journal of Machine Learning Research (743--746),"PyBrain",Volume 11, 2010

συστημάτων όπου ακόμη και η επιλογή της γλώσσας προγραμματισμού έχει να συμβάλει σε ταχύα ανάπτυξη ο χρόνος είναι μια σημαντική παράμετρος που επηρρέασε την επιλογή κατασκευής Απλού Πράκτορα.

Σε αντιπαραβολή αξίζει να αναφερθεί ότι η ανάπτυξη αρχικής διεπαφής μέσω PyGame διαρκεί μερικά μόλις λεπτά, η κατασκευή διαχείριση και αναζήτηση ενός γράφου μέσω NetWorkX χρειάζεται το πολύ 1 ώρα για ένα προγραμματιστή ικανοποιητικά εξοικειωμένο με την Python. Η ανάπτυξη ενός 2-player παιχνιδιού, βέβαια, απαιτεί αρκετά μεγαλύτερο χρόνο στην κατασκευή του.

4.3 Συμπεράσματα κύκλου ανάπτυξης

Κατά την διάρκεια ανάπτυξης του Συστήματος της Μεταπτυχιακής Διατριβής, υπήρξαν αρκετοί επανέλεγχοι και συνεχείς αναδράσεις από τα προβλήματα που προέκυπταν.
Κύριοι σταθμοί ήταν:

1. Ορισμός χρήσης σειριακών αρχείων καταγραφής καταστάσεων, κινήσεων και συμβάντων του παιχνιδιού - Δεκ 2011
2. Απόρριψη έτοιμων συστημάτων Ε.Μ όπως PyBrain και RL-Glue - Μαρ 2012
3. Επιλογή σταθμισμένου δίγραφου ώς δομή - Μάιος 2012

4.4 Προτεινόμενες Επεκτάσεις

Το σύστημα μπορεί να επεκταθεί στους 3 τομείς του Παιχνίδι/Πράκτορας/Δοκιμές ώς εξής:

Πρωτόκολλο Ενισχυτικής Μάθησης Πολυπρακτορικών Συστημάτων

Υπάρχουσες υλοποιήσεις λογισμικού Ενισχυτικής Μάθησης είναι δύσκολο να ενσωματωθούν εύκολα σε οποιοδήποτε πρόβλημα. Με τον ορισμό ενός ανοικτού πρωτοκόλλου που θα έχει σχεδιαστεί με την προοπτική χρήσης ενός ή περισσοτέρων πρακτόρων, περιβάλλοντων ή πειραμάτων θα είναι ευκολότερη η προσαρμογή σε ήδη υπάρχοντα συστήματα.

Τροποποίηση γεννήτριας τυχαίων κινήσεων σε παίκτη τυχαίων κινήσεων

Δημιουργείται έτσι ένας μηχανικός αντίπαλος εκπαίδευσης τόσο ανθρώπου όσο και του Πράκτορα. Θα πρέπει να λαμβάνει τον Ρόλο του Παίκτη 1 ή Παίκτη 2.

Τροποποίηση Πράκτορα ώστε να μπορεί να πάρει και το ρόλο του Παίκτη 1

Προς το παρόν ο Παίκτης 1 είναι Άνθρωπος και ο Πράκτορας είναι πάντα Παίκτης 2. Με αυτό τον τρόπο μπορούμε να έχουμε αντιπάλους Τυχαίο Πράκτορα με Πράκτορα ή Πράκτορα με Πράκτορα.

Διασύνδεση του πράκτορα με βιβλιοθήκη RL-Glue

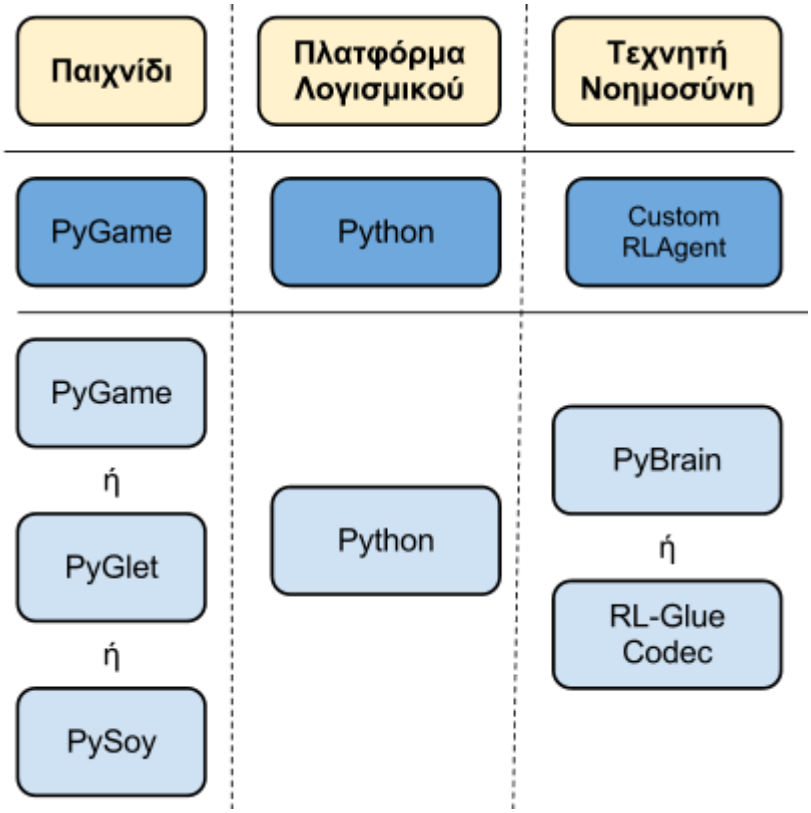
Υλοποιόντας τον Πράκτορα μέσα από το οικοσύστημα RL-Glue, το Παιχνίδι συνεχίζει και είναι υλοποιημένο σε Python, αλλά ο Πράκτορας μπορεί να κατασκευαστεί με μια πληθώρα εργαλείων[1].

[1]RL-Glue Codecs: C/C++, Java, Lisp, Matlab, Python

Κεφάλαιο 5

Επίλογος

Η Python και οι βιβλιοθήκες που διαθέτει αποτελούν ένα ισχυρό εργαλείο στην ανάπτυξη συστημάτων παιγνίων και τεχνητής νοημοσύνης.



Εικόνα: Διασύνδεση Τεχνολογιών για κατασκευή παιγνίων και πρακτόρων ενισχυτικής μάθησης

Μέσα από αυτή την Μεταπτυχιακή Διατριβή έγινε προσπάθεια αναζήτησης και τυποποίησης μεθόδων συνδυασμού των δύο με κοινό τόπο την συνιστώσα της γλώσσας προγραμματισμού Python, καθώς και η καταγραφή υπαρχουσών τεχνολογιών και πρόταση μιας αρχικής μεθόδου ανάπτυξης συνδυαστικού συστήματος παιγνιδιού με τεχνητή νοημοσύνη.

Επίσης αξίζει να σημειωθεί ότι η συνεχής τεκμηρίωση, βοήθησε στην σταδιακή ανάπτυξη του συστήματος καθώς και σε ορισμό της αρχιτεκτονικής απλών ή και σύνθετων παιγνίων με χρήση τεχνητής νοημοσύνης και ιδιαίτερα της Ενισχυτικής Μάθησης.

Παράρτημα Α

Αποσπάσματα κώδικα

```
1.      def getRandAction(columns):
2.          a = random.randint(0, 3)
3.          if(a==0 or a==1):
4.              c = random.randint(0, columns-1)
5.          else:
6.              c=-1
7.          return a,c
```

Κώδικας Α.1: Συνάρτηση τυχαίων κινήσεων

```
1.          def setRandAction(self, Game, Move):
2.              p1, p2, wl = self.arrays_load_from_txt_p1p2wl_int(Game, Move)
3.              f = Functions()
4.              a = random.randint(0, 3)
5.              if(a==0 or a==1):
6.                  c = random.randint(0, self.get_columns_number()-1)
7.              else:
8.                  c=-1
9.
10.             cmd = self.setAction(a,c)
11.             exec(cmd[0])
12.             reward = GameState().reward(wl, p1 , p2)
13.             if(reward >=0):
14.                 return cmd
15.             else:
```

```
16.             print "\t\t\tTrying another random action..."
17.             return self.setRandAction(Game, Move)
```

Κώδικας Α.2: Συνάρτηση τυχαίων κινήσεων χωρίς κίνδυνο στον Πράκτορα

```
1.      WATER_LEVEL = np.ones((WL_HEIGHT,WL_WIDTH), dtype=np.bool)
2.      BOATS_ARRAY = np.array([np.invert(WATER_LEVEL), np.invert(WATER_LEVEL)],
dtype=bool)
```

Κώδικας Α.3: Ορισμός διαστάσεων νερού και αρχικών θέσεων σκαφών μέσω NumPy

```
1.          def boat_over_water(self, water_level, boat):
2.              try:
3.                  and_arrays = water_level & boat
4.                  num = len(np.where(and_arrays==True)[0])
5.                  if(num>0):
6.                      return True
7.                  else:
8.                      return False
9.              except Exception as e:
10.                 print "GameState.boat_over_water error"
```

Κώδικας Α.4: Δοκιμή ύπαρξης νερού κάτω από σκάφος - προσοχή στην γραμμή 4

```
1.          def reward(self, wl, p1, p2):
2.              try:
3.                  wl = np.array(wl)
4.                  p1 = np.array(p1)
5.                  p2 = np.array(p2)
6.                  retval = 2
7.                  boats = (p1, p2)
8.                  xy0 = self.get_coords(boats[0])
9.                  xy1 = self.get_coords(boats[1])
10.                 for i in range(0,2):
11.                     if self.boat_over_water(wl, boats[i])==False:
12.                         retval = i
13.                         break
14.                 else:
15.                     try:
```

```
16.                 Y,X = self.get_coords(boats[i])
17.                 Y=Y-1
18.                 if(Y>=0):
19.                   if wl[Y][X]==True:
20.                     retval = i
21.                     break
22.               except:
23.                 pass
24.           if (retval == 0):
25.             retval = 1
26.           elif (retval == 1):
27.             retval = -1
28.           else:
29.             retval = 0
30.           if(np.array_equal(p1, df.BOAT_ARRAY_TARGET[0])):
31.             retval = -1
32.           if(np.array_equal(p2, df.BOAT_ARRAY_TARGET[1])):
33.             retval = 1
34.           return retval
35.         except Exception as e:
36.           print e
37.           return 0
```

Κώδικας Α.5: Συνάρτηση GameState κλάσσης για αμοιβή σε συγκεκριμένη κατάσταση s_{wl}, s_{p1}, s_{p2}. Στην γραμμή 30 και 32 γίνεται σύγκριση για το εαν το σκάφος είναι στην αρχική θέση του αντίπαλου καρνάγιου.

Παράρτημα Β

Δενδρική Δομή Έργου

```
svn checkout https://rltankattack.googlecode.com/svn/ rltankattack
```

rltankattack

```
|
+---code
|   \---RLTankAttack System Source Code In Python
|       |   .project
|       |   .pydevproject
|       |   Boat.py
|       |   DEFINES.py
|       |   Earth.py
|       |   Functions.py
|       |   GameState.py Game State Class: Calculates reward based on current state
|       |   Message.py
|       |   PlotEngine.py
|       |   RLAgent.py RLTankAttack Player 2 Agent: uses LearnedFromGame function at the end of each game
|       |   RLAgentBrown.py
|       |   RLAgentSmith.py Starts Agent as player 2
|       |   RLGraphBuilder.py Graph Builder: Iterative
|       |   RLGraphBuilder_v2.py Graph Builder: Random
|       |   RLGraphShow.py
|       |   RLTANKATTACK.py Control Panel for the System
|       |   RunGame.py Start the Game
|       |   Speak.py
|       |   test.py
|       |   Valve.py Valve Sprite
|       |   Water.py Water Sprite
|       |   WaterLevel.py
|       |   WhosTurn.py
```

```
|       |   WLSetSize.py
|       |
|       +---.settings
|       |       org.eclipse.core.resources.prefs
|       |
|       +---games Holds Directories with games, moves, states and actions
|       +---graphs
|       |   |   RLGraph.py Graph Handler Class
|       |   |   RLGraphActionsShow.py
|       |   |   RLGraphBuilder.gpickle Pickled RLGraph Class
|       |   |   RLGraphRewardShow.py
|       |   |   __init__.py
|       |   |
|       |   +---dots Dot Files for Graphs
|       |   +---images
|       |   \---reports Graph Reports/b>
|       +---images Game Images
|       |
|       +---py2exe
|       |
|       +---readme
|       |       actions.html
|       |       README-Python.txt
|       |       README.txt
|       |
|                  +---settings   Settings   Directory:   Plot   Engine   to   use
(Matplotlib/GraphViz), Water Level Size
|       |       plot_engine
|       |       wl_size
|       |
|       \---tests Test playground
|               animate_plot.py
|               graph.png
|               graphviz.py
|               pybrain_test.py
|
+---documents Documents and Various notes for the Thesis
|   +---mindmap
|   |       GAME-Flow.dia
|   |       GAME-Flow.png
|   |       GamesGraph.png
```

```
|   |       GAME_RLTA_XML.png
|   |       Game_RLTA_XML_EXAMPLE.xml
|   |       Overlay-MatPlotLib.png
|   |       PythonAndLibs.png
|   |       PythonStackDependencies.gv  Requires GraphViz
|   |       RLGraphActions.png
|   |       RLGraphAfterLearning5Games.png
|   |       RLGraphRewards.png
|   |       RLTANKATTACKUsageGraph.png
|   |       RLTANKATTACK_CONTROL_PANEL.png
|   |       RLTankAttack_MindMap.mm  Requires FreeMind
|   |       Stack.dia
|   |       Stack.png
|   |       START-to-finish.png
|   |       thegame.png
|   |       ThoughtsOnClicking.png
|   |       ThoughtsOnClicking_v2.png
|   |
|   +---thesis Thesis Directory: PDF+PPTX
|   |   |   README.txt
|   |   |   Spiros_Gezerlis_OUC.AC.CY_IS_MSc_Thesis_gr.docx
|   |   |   Spiros_Gezerlis_OUC.AC.CY_IS_MSc_Thesis_gr.pdf
|   |   |
|   |   \---Presentation
|   |       |   Presentation.pdf
|   |       |   Presentation.pptx
|   |       |
|   |       \---png
|   |
|   +---videos  Video ScreenCasts
|   |   +---Demos
|   |   |
|   |   \---Old
|   |
|   \---win32 Requirements
|           Graphviz 2.28.zip GraphViz - unzip to c:\Program Files (x86)
|           Python27.zip Python containing libs - unzip to c:\
|
\---wiki
```

Βιβλιογραφία

[1] Jennings & Wooldridge, 18 Jan 1996, "Software Agents" IEE Review
[2] Sutton Barto, "Reinforcement Learning: An Introduction", 1998
[3] OSKAR ARVIDSSON, LINUS WALLGREN, "Q-Learning for a Simple Board Game", Sweden 2010 http://www.csc.kth.se/utbildning/kandidatexjobb/datateknik/2010/rapport/arvidsson_oskar_OCH_wallgren_linus_K10047.pdf
[4] Brian Tanner and Adam White. RL-Glue: Language-Independent Software for Reinforcement-Learning Experiments. Journal of Machine Learning Research, 10(Sep):2133--2136, 2009. (BibTex)
[5] Schaul, Tom & Bayer, Justin & Wierstra, Daan & Sun, Yi & Felder, Martin & Sehnke, Frank & Ruckstiess, Thomas & Schmidhuber, Jurgen, "Journal of Machine Learning Research (743--746),"PyBrain",Volume 11, 2010