ΤΕΙ Δυτικής Μακεδονίας –Πανεπιστήμιο Μακεδονίας

ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

Μέθοδοι αναζήτησης και εφαρμογή τους για την επίλυση λαβύρινθων

ΜΑΡΔΑΝΗΣ ΔΗΜΗΤΡΙΟΣ

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:

Δρ. ΟΡΟΒΑΣ ΧΡΗΣΤΟΣ

Κοζάνη 2006

Η εργασία αυτή αφιερώνεται στις δύο καλές νεράιδες της ζωής μου, που πίστεψαν σε μένα, μου έδιναν κουράγιο και ήταν δίπλα μου σε κάθε στιγμή, την μητέρα μου Γεωργία και την κοπέλα μου Νίκη.

Copyright © Μαρδάνης Δημήτριος, 2006
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΕΧΟΜΕΝΑ 3

ΠΕΡΙΛΗΨΗ 5

SUMMARY 6

ΚΕΦΑΛΑΙΟ 1 7

ΕΙΣΑΓΩΓΗ 7

1.1 Γενικά 8
1.2 Δομή της εργασίας 10
ΚΕΦΑΛΑΙΟ 2 11

ΕΠΙΛΥΣΗ ΠΡΟΒΛΗΜΑΤΩΝ ΜΕ ΑΝΑΖΗΤΗΣΗ 11

2.1 Εισαγωγή 11
2.2 «Έξυπνοι» πράκτορες 12
2.3 Μια ματιά στο παρελθόν 14
2.4 Καθορισμός προβλήματος και λύσης 21
2.5 Δόμηση και κατανόηση του προβλήματος 23
2.6 Εύκολα Προβλήματα 24
2.6.1 Πρόβλημα «ηλεκτρικής σκούπας» 24
2.6.2 Παζλ 8 Θέσεων 25
2.6.3 ν – Βασίλισσες 28
2.7 Προβλήματα στον Πραγματικό Κόσμο 29
2.8 Αναζήτηση κατά Πλάτος 30
2.9 Αναζήτηση κατά Βάθος 32
2.10 Αναζήτηση σε ορισμένο βάθος 34
2.11 Αναζήτηση βέλτιστης επέκτασης 34
2.12 Αναζήτηση με τον αλγόριθμο Α* 34
2.13 Κορυφαίες στιγμές της Τ.Ν. 41
2.13.1 Διαστημικά προγράμματα 41
2.13.2 Επίλυση εύκολων προβλημάτων 41
2.13.3 Αυτόνομος έλεγχος 42
2.13.4 Διαγνωστικά συστήματα 42
2.13.5 Ρομποτική 42
2.14 Συμπερασματικά 42
ΚΕΦΑΛΑΙΟ 3 44

Η ΕΦΑΡΜΟΓΗ ΚΑΙ Ο ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΤΗΣ 44

3.1 Εισαγωγή 45
3.2 Ο αλγόριθμος κίνησης των κινητών πρακτόρων 45
3.3 Το interface του λαβύρινθου 48
3.4 Η εκκίνηση 49
3.5 Το μενού «Νέος λαβύρινθος» 50
3.6 Το μενού «Μηδενισμός» 51
3.7 Το μενού «Exit» 52
3.8 Το μενού «Τεχνητή Νοημοσύνη – Εκκίνηση» 53
3.9 Το μενού «Τεχνητή Νοημοσύνη – Χάρτης ΤΝ» 54
3.10 Το μενού «Χάρτης Περασμάτων» 55
3.11 Ο σχεδιασμός και ο κώδικας της εφαρμογής 57
3.11.1 Οι έλεγχοι για την κίνηση των πρακτόρων 61
3.11.2 Τερματισμός του παιχνιδιού 66
3.12 Συναρτήσεις της εφαρμογής 68
3.12.1 Οι συναρτήσεις lookright & lookleft 69
3.12.2 Η συνάρτηση counttimes( ) 71
3.12.3 Η συνάρτηση showcounter( ) 71
3.12.4 Η συνάρτηση trace( ) 72
3.13 Προβλήματα που αντιμετωπίστηκαν 72
3.13.1 Μέγεθος του λαβύρινθου 72
3.13.2 Εγκλωβισμός του παίκτη 72
3.13 3 Ταχύτητα βρόγχων 74
3.13.4 Κίνηση στην φόρμα About 74
3.13.5 Δυναμική ενημέρωση του πίνακα μέτρησης περασμάτων 75
3.13.6 Εμφάνιση ίχνους 76
3.14 Συμπέρασμα 77
ΚΕΦΑΛΑΙΟ 4 79

ΣΥΜΠΕΡΑΣΜΑΤΑ – ΠΡΟΟΠΤΙΚΕΣ 79

ΠΑΡΑΡΤΗΜΑ ..... .80
ΒΙΒΛΙΟΓΡΑΦΙΑ 123

ΠΕΡΙΛΗΨΗ

Ο λαβύρινθος που χρησιμοποιεί τεχνητή νοημοσύνη για την επίλυσή του, δημιουργήθηκε αρχικά για τις ανάγκες του μαθήματος της τεχνητής νοημοσύνης του μεταπτυχιακού τμήματος της επιχειρηματικής πληροφορικής και στηρίζεται στο πρόβλημα που παρουσιάζεται στο κεφάλαιο 4 του βιβλίου των Stuart Russell & Peter Norvig, 2003,Artificial Intelligence-A Modern Approach,Second Edition, Prentice Hall όπου ένας έξυπνος πράκτορας πρέπει να βρει ένα θησαυρό αποφεύγοντας ένα τέρας (Wumpus). Χτίστηκε στην πλατφόρμα προγραμματισμού VISUAL BASIC 6.00. Η έκδοση του λαβύρινθου που υλοποιείται στην συγκεκριμένη εργασία αποτελεί εξέλιξη της αρχικής μορφής και περιλαμβάνει εκτός από τον παίκτη ο οποίος προσπαθεί να φτάσει στην έξοδο και το τέρας το οποίο προσπαθεί να ανακόψει την πορεία του παίκτη προς αυτήν. Ο παίκτης αποτελεί τον κινητό πράκτορα ο οποίος κάνει χρήση ενός υβριδικού αλγόριθμου αναζήτησης. Ο αλγόριθμος αυτός κάνει χρήση αναζήτησης έχοντας όμως την ικανότητα να συγκρατήσει στην μνήμη του πόσες φορές πέρασε από κάποιο σημείο. Έτσι μπορεί να αναγνωρίσει αν έχει «κλειστεί» και δεν μπορεί να ξεφύγει οπότε και αλλάζει πορεία. Αντίθετα το τέρας, που αποστολή του είναι να βρει και να ανακόψει την πορεία του παίκτη, αποτελεί το κινητό εμπόδιο μέσα στο λαβύρινθο, το οποίο κινείται χωρίς να χρησιμοποιεί κάποια μνήμη, αλλά κάνει χρήση τυφλής αναζήτησης και πιο συγκεκριμένα αναζήτησης εις πλάτος.

SUMMARY

The labyrinth that uses artificial intelligence for his resolution, was created initially for the needs of the course of artificial intelligence of postgraduate department of enterprising information technology and is supported in the problem that is presented in capital 4 of Stuart Russell & Peter Norvig, 2003,Artificial Intelligence-A Modern Approach,Second Edition, Prentice Hall where an intelligent agent should find a treasure avoiding a monster (Wumpus). It was programmed based on the platform of programming VISUAL BASIC 6.00. The version of the labyrinth that is materialized in the particular project constitutes development of the initial form and includes apart from the player which tries to reach the exit and the monster which tries to stop the course of the player to the exit. The player constitutes the mobile agent which makes use of a hybrid searching algorithm. This algorithm makes use of search having however the faculty to retain in the memory how much times it passed from some point. Thus it can recognize if "it has been closed" and cannot escape therefore also change course. On the contrary the monster, that its mission is to find and stop the course of player, constitutes the mobile obstacle in the labyrinth, which is moved without using any kind of memory, but makes use of blind search and more concretely search to width.

ΚΕΦΑΛΑΙΟ 1

ΕΙΣΑΓΩΓΗ

1.1 Γενικά

Η Τεχνητή Νοημοσύνη αποτελεί έναν από τους πιο γρήγορα αναπτυσσόμενους τομείς της πληροφορικής. Από τις αρχές της δεκαετίας του 1950 οι επιστήμονες προσπαθούν να δημιουργήσουν συστήματα αυτοματισμού. Τα συστήματα αυτά ξεκίνησαν έχοντας σαν στόχο τους να επιλύσουν απλά προβλήματα τα οποία έχουν την μορφή παιγνίων (όπως το σκάκι) και σαν τελική μορφή του στόχου τους είναι η εξομοίωση της διαδραστικότητας ενός πράκτορα σε πραγματικό περιβάλλον, ο οποίος θα κινείται και θα συμπεριφέρεται σαν ένα ανθρώπινο ον. Συγκεκριμένα σαν Τεχνητή Νοημοσύνη ορίζεται η ικανότητα που επιδιώκεται να αποκτήσουν οι ηλεκτρονικοί υπολογιστές και τα συστήματα αυτοματισμού που καθοδηγούνται από ηλεκτρονικό υπολογιστή, ώστε να διεκπεραιώνουν πράξεις και εργασίες τις οποίες ο άνθρωπος εκτελεί χάρη στη νοημοσύνη του, όπως για παράδειγμα η κατανόηση της γλώσσας και των νοημάτων, η αξιολόγηση στοιχείων και λήψη αποφάσεων καθώς επίσης και η ικανότητα μάθησης (Επιστήμη και Ζωή, τόμος 22, σελ 240, άρθρο: Τεχνητή Νοημοσύνη). Στη συγκεκριμένη εργασία θα αναλυθούν οι αλγόριθμοι αναζήτησης (κατά βάθος, κατά πλάτος καθώς και υβριδικοί αλγόριθμοι), θα περιγράψουμε την επίλυση παιγνίων και θα αναλύσουμε μια εφαρμογή η οποία έχει δημιουργηθεί ακριβώς για να μελετηθούν οι αλγόριθμοι αναζήτησης. Η εφαρμογή που δημιουργήθηκε είναι η προσομοίωση ενός λαβυρίνθου.

Ο λαβύρινθος που χρησιμοποιεί τεχνητή νοημοσύνη για την επίλυσή του, δημιουργήθηκε αρχικά για τις ανάγκες του μαθήματος της τεχνητής νοημοσύνης του μεταπτυχιακού τμήματος της επιχειρηματικής πληροφορικής και στηρίζεται στο πρόβλημα που παρουσιάζεται στο κεφάλαιο 4 όπου ένας έξυπνος πράκτορας πρέπει να βρει ένα θησαυρό αποφεύγοντας ένα τέρας (Wumpus). Χτίστηκε στην πλατφόρμα προγραμματισμού VISUAL BASIC 6.00. Η έκδοση του λαβύρινθου που υλοποιείται στην συγκεκριμένη εργασία αποτελεί εξέλιξη της αρχικής μορφής και περιλαμβάνει εκτός από τον παίκτη ο οποίος προσπαθεί να φτάσει στην έξοδο και το τέρας το οποίο προσπαθεί να ανακόψει την πορεία του παίκτη προς αυτήν. Ο παίκτης αποτελεί τον κινητό πράκτορα ο οποίος κάνει χρήση ενός υβριδικού αλγόριθμου αναζήτησης. Ο αλγόριθμος αυτός κάνει χρήση αναζήτησης έχοντας όμως την ικανότητα να

συγκρατήσει στην μνήμη του πόσες φορές πέρασε από κάποιο σημείο. Έτσι μπορεί να αναγνωρίσει αν έχει «κλειστεί» και δεν μπορεί να ξεφύγει οπότε και αλλάζει πορεία. Αντίθετα το τέρας, που αποστολή του είναι να βρει και να ανακόψει την πορεία του παίκτη, αποτελεί το κινητό εμπόδιο μέσα στο λαβύρινθο, το οποίο κινείται χωρίς να χρησιμοποιεί κάποια μνήμη, αλλά κάνει χρήση τυφλής αναζήτησης και πιο συγκεκριμένα αναζήτησης εις πλάτος. Βεβαίως θα αναφερθούμε εκτενέστερα στην κίνηση του παίκτη και του κινητού εμποδίου παρακάτω.

Τα βασικά χαρακτηριστικά του λαβύρινθου είναι τα εξής:

- Με επιλογή του ο χρήστης μπορεί να δημιουργήσει έναν νέο τυχαίο λαβύρινθο.
- Μπορεί να επιλυθεί σε οποιαδήποτε τυχαία μορφή του, αν έχει λύση.
- Υπάρχει ο κινητός πράκτορας του οποίου σκοπός είναι να βρει την έξοδο.
- Υπάρχει κινητό εμπόδιο το οποίο ανακόπτει την πορεία του παίκτη.
- Ο αλγόριθμος που ελέγχει τον κινητό πράκτορα κάνει χρήση μνήμης.

Η τρέχουσα έκδοση του λαβύρινθου περιλαμβάνει και κάποια επιπλέον χαρακτηριστικά. Τα χαρακτηριστικά αυτά προστέθηκαν για να γίνει πιο ενδιαφέρον ο λαβύρινθος από πλευρά παρατηρητικότητας, καθώς επίσης και για να επιδειχθεί η λειτουργία της τεχνητής νοημοσύνης.

Επιπλέον χαρακτηριστικά:

- Το τέρας WUMPUS το οποίο κινείται επίσης από την τεχνητή νοημοσύνη του λαβύρινθου.
- Τυχαία επιλογή θέσης για τον παίκτη, την έξοδο και το WUMPUS.
- Ο κινητός πράκτορας αφήνει ίχνη σε κάθε βήμα που κάνει για να αναδειχθεί η πορεία που ακολούθησε.
- Μουσική στο παρασκήνιο.
- Ατμοσφαιρικό interface.
- Πίνακας τεχνητής νοημοσύνης που καταγράφει και δείχνει το τι έχει μάθει ο κινητός πράκτορας.
- Πίνακας ο οποίος μετράει τις φορές που προσπέλασε το κάθε σημείο του λαβύρινθου ο κινητός πράκτορας.
- Δυνατότητα επιλογής να δει ο χρήστης κάποιες πληροφορίες και σημειώσεις για τον προγραμματισμό του λαβύρινθου καθώς και φωτογραφικό υλικό.

1.2 Δομή της εργασίας

Στο δεύτερο κεφάλαιο θα αναφερθούν οι αλγόριθμοι αναζήτησης που χρησιμοποιούνται σήμερα. Οι αλγόριθμοι αυτοί είναι η αναζήτηση κατά βάθος, η αναζήτηση κατά πλάτος, και υβριδικοί αλγόριθμοι που συνδυάζουν τους δύο τελευταίους. Επίσης έχουν υλοποιηθεί αλγόριθμοι που χρησιμοποιούν κάποια μνήμη για να μπορούν να καταγράφουν τα σημεία που προσπέλασαν και πόσες φορές τα προσπέλασαν και έχουν αναπτυχθεί ακόμη και αλγόριθμοι που χρησιμοποιούν πολλούς πράκτορες για να ανιχνεύουν τους γύρω κόμβους έτσι ώστε ο κεντρικός πράκτορας να μπορεί να επιλέξει έπειτα την καλύτερη κίνηση προς τη λύση.

Στο τρίτο κεφάλαιο θα αναλυθεί ο υβριδικός αλγόριθμος που σχεδιάστηκε και υλοποιήθηκε για τις ανάγκες τις συγκεκριμένης εργασίας. Γενικά, αποτελεί συνδυασμό δύο αλγορίθμων αναζήτησης. Του αλγόριθμου αναζήτησης κατά βάθος, με στόχο να φτάσει στην έξοδο όσο το δυνατό πιο γρήγορα και ενός αλγόριθμου που κάνει χρήση μνήμης για να μπορέσει να «θυμάται» τα βήματα που έκανε και να βοηθηθεί για να μην κολλήσει σε κάποιο δύσκολο σημείο του λαβύρινθου. Ο αλγόριθμος που ελέγχει την κίνηση του κινητού εμποδίου είναι αλγόριθμος τυφλής αναζήτησης κατά πλάτος.

Επίσης στο τρίτο κεφάλαιο περιγράφεται και αναλύεται η εφαρμογή του λαβύρινθου. Περιγράφεται το interface της εφαρμογής, η υλοποίησή της, παρατίθενται τα εικονίδια που χρησιμοποιήθηκαν, καθώς και γενικά η όλη νοοτροπία που επηρέασε την υλοποίηση της εφαρμογής. Αναλύεται ο κώδικας, καθώς και περιγράφονται οι τρόποι που επινοήθηκαν για να ξεπεραστούν τα προβλήματα που προέκυψαν κατά την υλοποίηση της εφαρμογής. Τέλος, αναφορά γίνεται και σε κάποιες λεπτομέρειες που δεν επηρεάζουν άμεσα την εφαρμογή, όπως η φόρμα «about», όπου περιέχει κάποιες σκέψεις και σημειώσεις του προγραμματιστή της εφαρμογής.

ΚΕΦΑΛΑΙΟ 2

ΕΠΙΛΥΣΗ ΠΡΟΒΛΗΜΑΤΩΝ ΜΕ ΑΝΑΖΗΤΗΣΗ

2.1 Εισαγωγή

Στο πρώτο κεφάλαιο ,το οποίο αποτελούσε και την εισαγωγή της εργασίας, είδαμε το σύνολο των χαρακτηριστικών της εφαρμογής. Είδαμε τις βασικές απαιτήσεις καθώς επίσης και τα επιπλέον χαρακτηριστικά που προστέθηκαν μετέπειτα. Σε γενικές γραμμές περιγράψαμε την δομή της εργασίας καθώς και τι θα αναφέρουμε στο κάθε ένα κεφάλαιο.

Στο δεύτερο κεφάλαιο θα αναλύσουμε τους «έξυπνους» πράκτορες, θα αναφερθούμε στο παρελθόν και την εξέλιξη της Τ.Ν., και θα κάνουμε έναν καθορισμό του προβλήματος, καθώς και της λύσης για να μπορέσουμε να αναπτύξουμε τους κατάλληλους αλγορίθμους. Έπειτα θα αναφερθούμε στην δομή του προβλήματος και θα κάνουμε μια εκτενή ανάλυση όσον αφορά τα προβλήματα παιγνίων. Στη συνέχεια θα μελετήσουμε τα προβλήματα που θα πρέπει να ξεπεράσει κάποιος πράκτορας σε περιβάλλον πραγματικό και τέλος θα αναλύσουμε τους αλγορίθμους αναζήτησης και συγκεκριμένα την αναζήτηση κατά πλάτος, την αναζήτηση σε βάθος, την αναζήτηση σε ορισμένο βάθος και τελικά έναν πιο εξελιγμένο αλγόριθμο αναζήτησης, τον A* ο οποίος εισάγει την έννοια του κόστους, πριν αποφασίσει την διαδρομή που θα ακολουθήσει για να βρει την βέλτιστη λύση.

2.2 «Έξυπνοι» πράκτορες

Οι «έξυπνοι» πράκτορες υποτίθεται ότι μεγιστοποιούν το ποσοστό επιτυχίας κατά τη διάρκεια επίλυσης κάποιου προβλήματος. Πολλές φορές το να καταφέρει να έχει κάποιος πράκτορας το μέγιστο αποτέλεσμα είναι πολύ εύκολο καθώς μπορεί πολύ απλά να εντοπίσει επακριβώς το στόχο του και να στοχεύσει στο να τον ικανοποιήσει. Για να κατανοήσουμε το πώς ένας πράκτορας μπορεί να λειτουργήσει όπως αναφέρθηκε παραπάνω θα κάνουμε ένα μικρό παράδειγμα.

Ας υποθέσουμε ότι κάποιος τουρίστας έρχεται στην Ελλάδα για να δει τα αξιοθέατα της χώρας μας. Έχει πολλές επιλογές και πρέπει να πάει σε πολλές πόλεις για να δει τα αρχαία. Οι πόλεις που μπορεί να επισκεφτεί είναι: Αθήνα, Θεσσαλονίκη, Λάρισα, Ολυμπία, Μυκήνες, Βεργίνα, Δελφοί. Ο τουρίστας υποθέτουμε ότι έχει ένα εισιτήριο για να επιστρέψει στη χώρα του από την Αθήνα. Βάσει αυτού ο πράκτορας θα πρέπει να υπολογίσει ότι η τελευταία πόλη που θα επισκεφτεί θα είναι η Αθήνα και θα πρέπει ταυτόχρονα να έχει επισκεφτεί όσο το δυνατό περισσότερες πόλεις, αν

όχι όλες, στο χρονικό διάστημα που του μένει έως ότου λήξει το εισιτήριό του. Σ' αυτό το στάδιο έχουμε περιορίσει τις επιλογές του πράκτορα έτσι ώστε να αναγκαστεί να επισκεφτεί την Αθήνα στο τέλος. Η κατανόηση του στόχου είναι και το πρώτο βήμα στο να επιλυθεί το πρόβλημα.

Εφόσον έχει κατανοηθεί ο στόχος θα πρέπει να θέσουμε κάποιους κανόνες έτσι ώστε να μπορέσουμε να φτάσουμε σ' αυτόν. Η αποστολή του πράκτορα είναι να βρει την καταλληλότερη σειρά ενεργειών με την οποία θα επιτευχθεί ο στόχος. Πριν από αυτό θα πρέπει να αποφασίσει ο πράκτορας ποιες ενέργειες θα πρέπει να λάβει υπόψη του. Για παράδειγμα, αν θα έπρεπε να υπολογιστούν ενέργειες όπως "μετακίνησε το αριστερό πόδι μπροστά μισό μέτρο" ή "στρίψε το τιμόνι 14 μοίρες δεξιά", τότε το πιο πιθανό είναι ότι ο πράκτορας δεν θα μπορούσε ούτε να βάλει μπροστά το αυτοκίνητό του, πόσο μάλλον να φτάσει στην Αθήνα, ακριβώς γιατί το επίπεδο της λεπτομέρειας είναι πάρα πολύ μεγάλο σε σχέση με το υπόλοιπο περιβάλλον με το οποίο έρχεται σε επαφή ο πράκτορας, με αποτέλεσμα να υπάρχουν πάρα πολλά βήματα μέχρι να βρεθεί λύση. Για τις ανάγκες της κατανόησης του παραδείγματος υποθέτουμε ότι ο πράκτορας μπορεί ανά πάσα στιγμή να κινηθεί ανάμεσα στις πόλεις. Υποθέτουμε επίσης ότι ο πράκτοράς μας ξεκινάει από τη Βεργίνα.

Σ' αυτό το στάδιο ο πράκτοράς μας γνωρίζει το στόχο ο οποίος είναι το να φτάσει οδηγώντας στην Αθήνα και το πρόβλημά του είναι η διαδρομή που θα ακολουθήσει από τη Βεργίνα. Έχει δύο επιλογές: να φύγει για Θεσσαλονίκη ή για Λάρισα. Καμία από τις δύο αυτές επιλογές δεν ικανοποιεί το στόχο. Εκτός και αν ο τουρίστας έχει ξαναέρθει Ελλάδα και ξέρει τις αποστάσεις, δεν θα ξέρει ποια διαδρομή να ακολουθήσει. Με άλλα λόγια ο πράκτορας δεν θα γνωρίζει ποια από τις δύο επιλογές είναι η συντομότερη οπότε θα πρέπει και να αποφασίσει στην τύχη. Για να μπορέσουμε να βοηθήσουμε τον τουρίστα μας, υποθέτουμε ότι έχει μαζί του έναν χάρτη ο οποίος γράφει τις χιλιομετρικές αποστάσεις των πόλεων. Ο πράκτορας μπορεί να χρησιμοποιήσει τις πληροφορίες που του παρέχει ο χάρτης και να κάνει ένα υποθετικό ταξίδι σε καθεμία από τις δύο πόλεις υπολογίζοντας έτσι την καλύτερη δυνατή διαδρομή έως ότου φτάσει στην Αθήνα. Σε γενικές γραμμές δηλαδή ο πράκτορας ο οποίος έχει αρκετές επιλογές άγνωστης όμως αξίας μπορεί να αποφασίσει τι ακριβώς να κάνει ερευνώντας πρώτα διαφορετικές δυνατές σειρές ενεργειών που οδηγούν στο στόχο και τέλος, αφού τις έχει ελέγξει όλες, μπορεί να επιλέξει την καλύτερη από αυτές.

Η διαδικασία με την οποία ο πράκτορας θα καταφέρει να βρει μια κατάλληλη σειρά ενεργειών για να φτάσει στη λύση του προβλήματος ονομάζεται έρευνα ή αλλιώς αναζήτηση. Ένας αλγόριθμος αναζήτησης δέχεται το πρόβλημα σαν είσοδο και η λύση που επιστρέφει είναι της μορφής μιας σειράς ενεργειών. Μόλις βρεθεί η λύση τότε οι ενέργειες που προτείνονται σαν λύση μπορούν να χρησιμοποιηθούν. Αυτή ονομάζεται φάση εκτέλεσης. Έτσι έχουμε σε γενικές γραμμές μία απλή δομή του τύπου «Δόμηση, Αναζήτηση, Εκτέλεση». Αφού δομηθεί το πρόβλημα και κατανοηθεί ο στόχος, ο πράκτορας καλεί τη διαδικασία αναζήτησης για να το λύσει. Έπειτα χρησιμοποιεί τη λύση για να καθοδηγήσει τις απαιτούμενες ενέργειες, κάνοντας οτιδήποτε προτείνει η λύση σαν το επόμενο σωστό βήμα και στη συνέχεια το αφαιρεί από τη σειρά των προγραμματισμένων ενεργειών. Όταν τελικά εκτελεστεί η λύση του προβλήματος, τότε ο πράκτορας θα ψάξει για να βρει ένα νέο στόχο.

2.3 Μια ματιά στο παρελθόν

Η πρώτη προσπάθεια που αναγνωρίστηκε γενικά σαν έρευνα πάνω στην Τεχνητή Νοημοσύνη έγινε από τους Warren(10), McCulloch και Walter Pitts (11). Οι τελευταίοι βασίστηκαν σε τρεις πηγές: στη γνώση της βασικής ψυχολογίας και λειτουργία των νευρώνων στον ανθρώπινο εγκέφαλο, σε μία ανάλυση της προτασιακής λογικής των Russel και Whitehead (12) και τέλος στην θεωρία των υπολογισμών του Turing. Οι McCulloch και Pitts (11) πρότειναν ένα μοντέλο τεχνητών νευρώνων όπου το κάθε νεύρο χαρακτηρίζεται σαν «ανοικτό» ή «κλειστό», με τον χαρακτηρισμό «ανοικτό» να ισχύει σαν αντίδραση στις διεγέρσεις από ένα συγκεκριμένο αριθμό γειτονικών νευρώνων. Με λίγα λόγια έδειξαν ότι οποιαδήποτε υπολογίσιμη συνάρτηση μπορεί να υπολογιστεί από μερικά δίκτυα συνδεδεμένων νευρώνων και ότι όλοι ο λογικοί σύνδεσμοι (and, or, not κλπ) μπορούσαν να εκφραστούν από τέτοια δίκτυα. Οι McCulloch και Pitts (11) επίσης υπέθεσαν ότι κατάλληλα καθορισμένα δίκτυα θα μπορούσαν ακόμη και να μάθουν. Ο Donald Hebb(13) (1949) επέδειξε έναν απλό αναβαθμιζόμενο ορισμό για την παραμετροποίηση των δυνάμεων σύνεσης μεταξύ νευρώνων. Ο ορισμός αυτός σήμερα αποκαλείται Hebbian learning και επηρεάζει τους ερευνητές και τα μοντέλα τους ακόμη και σήμερα.

Στο πανεπιστήμιο του Princeton φοίτησε και μία ακόμη μεγάλη μορφή στο χώρο της Τεχνητής Νοημοσύνης ο John McCarthy(14). Αφού αποφοίτησε πήγε στο Dartmouth College όπου έπεισε τους Minsky, Shannon και Rochester (14) να τον βοηθήσουν να συγκεντρώσουν όλους μαζί τους ερευνητές που υπήρχαν στην Αμερική και έκαναν εργασίες πάνω στα νευρωνικά δίκτυα, τις θεωρίες αυτοματισμού και την μελέτη της νοημοσύνης. Έτσι οργανώθηκε ένα μεγάλο ερευνητικό κέντρο το οποίου οι εργασίες έγιναν το καλοκαίρι του 1956.

Τα πρώτα χρόνια της ΤΝ ήταν γεμάτα από επιτυχίες αλλά σε περιορισμένη ακτίνα. Παίρνοντας υπόψη τους πρωτόγονους υπολογιστές και τα προγραμματιστικά εργαλεία που υπήρχαν εκείνη την εποχή και το γεγονός ότι οι υπολογιστές μέχρι και πριν από μερικά χρόνια δεν μπορούσαν να κάνουν τίποτα άλλο παρά αριθμητικές πράξεις θεωρούνταν εκπληκτικό όταν γινόταν κάτι «έξυπνο».

Η επιτυχία του LT ακολουθήθηκε από ένα νέο πρόγραμμα το General Problem Solver ή GPS. Το κύριο χαρακτηριστικό του όπως φάνηκε στην πράξη ήταν ότι όταν έλυνε κάποιο πρόβλημα, η σειρά με την οποία υπολόγιζε τις επιλογές που είχε ήταν παρόμοια με τον ανθρώπινο τρόπο σκέψης. Γι' αυτό και το GPS θεωρήθηκε σαν το πρώτο πρόγραμμα που εξομοίωνε τον ανθρώπινο τρόπο σκέψης. Η επιτυχία του προγράμματος αυτού ήταν τόσο μεγάλη που οδήγησε τους Allen (15) Newell και Herbert Simon (16), να διατυπώσουν την περίφημη Physical Symbol Hypothesis. Η θεωρία αυτή με λίγα λόγια λέει ότι οποιοδήποτε σύστημα με νοημοσύνη (μηχανή ή άνθρωπος) πρέπει να λειτουργεί υπολογίζοντας και επεξεργάζοντας μεγάλες δομές δεδομένων που να αποτελούνται από σύμβολα.

Στην IBM παράχθηκαν τα πρώτα προγράμματα ΤΝ. Ο Herbert Gelertner(17) (1959) δημιούργησε το Geometry Theorem Prover, το οποίο ήταν ικανό να αποδεικνύει θεωρήματα τη στιγμή που οι περισσότεροι σπουδαστές τα θεωρούσαν πολύ δύσκολο να αποδειχθούν. Το 1952 ο Arthur Samuel (18) ξεκίνησε και δημιούργησε μία σειρά προγραμμάτων για σκακιστές και έφτασε σε σημείο το πρόγραμμά του να παίζει καλύτερα από τον δημιουργό του. Το πρόγραμμά του δημοσιεύτηκε ακόμη και στην τηλεόραση τον Φεβρουάριο του 1956 κάνοντας πολύ μεγάλοι εντύπωση στο κοινό.

Το 1958 ο Marvin Minsky (19) πήγε στο MIT. Η συνεργασία του με τον McCarthy όμως δεν κράτησε. Ο Minsky (19) ακολούθησε διαφορετικό δρόμο από τον McCarthy (14). Ο τελευταίος χρησιμοποιούσε παρουσίαση και αιτιότητα σε επίσημη λογική, ενώ ο Minsky (19) επικεντρώθηκε στο να τρέχουν σωστά τα

προγράμματά του και τελικά ανάπτυξε ένα προφίλ εναντίον της λογικής. Το 1963 ο McCarthy(14) έκανε ένα εργαστήριο ειδικά για ΤΝ στο Stanford. Ο στόχος του ήταν να δημιουργήσει το υπέρτατο Advice Taker, το οποίο εξελίχθηκε βάσει ενός αλγόριθμου λογικής πρώτης τάξης, ο οποίος είχε σχεδιαστεί και αναπτυχθεί από τον Robinson (20).

Το 1963 οι σπουδαστές του Minsky(19) ανάπτυξαν προγράμματα που έλυναν περιορισμένα προβλήματα και έμειναν γνωστά σαν Microworlds. Ο James Slagle(21) ανέπτυξε το SAINT που μπορούσε να λύσει υπολογιστικά προβλήματα παρόμοια με αυτά που υπήρχαν στα μαθήματα του πρώτου έτους. Ο Tom Evans(22) (1968) ανάπτυξε το ANALOGY ένα πρόγραμμα που έλυνε προβλήματα αναλογίας στη Γεωμετρία που υπήρχαν στα τεστ IQ. Το1967 ο Daniel Bobrow(23) ανέπτυξε το επονομαζόμενο STUDENTπου μπορούσε να λύσει προβλήματα άλγεβρας όπως το παρακάτω:

> "Αν ο αριθμός των πελατών του Α είναι διπλάσιος του τετραγώνου του 20% του αριθμού των διαφημιστικών που έχει και ο αριθμός των διαφημιστικών είναι 45 ποιος είναι ο αριθμός των πελατών του Α."

Ο πιο δημοφιλής microworld ήταν ο κόσμος των τετραγώνων, ο οποίος αποτελείτο από ένα σετ από τετράγωνα τοποθετημένα σε μια οριζόντια επιφάνεια σαν τραπέζι. Ένα τυπικό πρόβλημα σ' αυτόν τον κόσμο είναι η ανασυγκρότηση των τετραγώνων με τέτοιο τρόπο που να μπορεί ο χρήστης να κινεί ένα τετράγωνο τη φορά.

Στην περίοδο της δεκαετίας του 1970 οι προγραμματιστές ήταν πάρα πολύ αισιόδοξοι. Σύμφωνα με τις προβλέψεις τους, σε 10 χρόνια από εκείνη την εποχή οι υπολογιστές θα ήταν πρωταθλητές στο σκάκι και ένα πολύ δύσκολο μαθηματικό θεώρημα θα μπορούσε να λυθεί από μία μηχανή. Οι προβλέψεις αυτές πραγματοποιήθηκαν σε 40 χρόνια αντί για 10. Αυτή η υπεραισιοδοξία οφειλόταν στην πάρα πολύ καλή απόδοση των συστημάτων ΤΝ σε απλά παραδείγματα. Παρόλα αυτά σχεδόν σε όλες τις περιπτώσεις τα συστήματα αυτά απέτυχαν παταγωδώς όταν χρησιμοποιήθηκαν σε πιο δύσκολα προβλήματα και με περισσότερες παραμέτρους.

Η πρώτη δυσκολία σημειώθηκε γιατί τα αρχικά προγράμματα περιείχαν ελάχιστη ή καθόλου γνώση των προβλημάτων που έλυναν, πετύχαιναν μόνο επειδή χρησιμοποιούνταν απλές συντακτικές λύσεις. Για παράδειγμα, το US National Research Council ήθελε να επιταχύνει τη διαδικασία μετάφρασης από τα ρώσικα στα αγγλικά των επιστημονικών άρθρων κατά την εκτόξευση του Sputnik το 1957. Αρχικά θεωρήθηκε ότι απλές συντακτικές μετατροπές βασισμένες στις γραμματικές

των αγγλικών και των ρωσικών και ταυτόχρονη αντικατάσταση λέξεων χρησιμοποιώντας ηλεκτρονικό λεξικό θα έφτανε για να δημιουργήσει το ακριβές νόημα των προτάσεων. Στην πραγματικότητα η μετάφραση απαιτεί γενική γνώση του αντικειμένου για να μπορέσει έτσι να δοθεί νόημα στην πρόταση. Οι διάσημες επαναμεταφράσεις του «το πνεύμα είναι πρόθυμο αλλά η σάρκα αδύναμη» αντικατοπτρίζει τη δυσκολία κατά τη μετάφραση. Τελικά το 1966 μία αναφορά από την αμερικάνικη επιτροπή δήλωνε ότι «δεν υπάρχουν μηχανές ικανές για μετάφραση γενικού επιστημονικού κειμένου και δεν θα υπάρξει καμία στο άμεσο μέλλον». Η αμερικανική κυβέρνηση αμέσως ακύρωσε όλους τους οικονομικούς πόρους για τα project ακαδημαϊκής μετάφρασης. Σήμερα οι μεταφράσεις μέσω μηχανών – προγραμμάτων είναι ένα ημιτελές μεν αλλά πολύ καλό εργαλείο για τεχνικά και διαφημιστικά έγγραφα δημοσιευμένα στο διαδίκτυο.

Το δεύτερο εμπόδιο ήταν η δυσκολία πολλών προβλημάτων που η ΤΝ προσπαθούσε να λύσει. Τα περισσότερα από τα πρώτα προγράμματα ΤΝ έλυναν τα προβλήματα δοκιμάζοντας διαφορετικούς συνδυασμούς βημάτων έως ότου βρισκόταν η λύση. Αυτή η στρατηγική δούλευε αρχικά γιατί οι μικρόκοσμοι περιείχαν πολύ λίγα αντικείμενα τα οποία απαιτούσαν πολύ λίγες πιθανές πράξεις οπότε υπήρχαν και πολύ λίγα υπολογιστικά βήματα μέχρι να βρεθεί η λύση. Στην αρχή πίστευαν ότι το να περάσουν σε μεγαλύτερα προβλήματα τα οποία θα μπορούσαν να λυθούν ήταν απλά ένα πρόβλημα γρηγορότερου hardware και μεγαλύτερης μνήμης. Βέβαια στο τέλος είδαν ότι το γεγονός ότι ένα πρόγραμμα μπορεί να βρει μία λύση δεν σημαίνει ότι το ίδιο πρόγραμμα περιέχει τους κατάλληλους μηχανισμούς για να βρει τη λύση και με πραγματικά δεδομένα.

Η τρίτη δυσκολία ήταν σε μερικούς στοιχειώδεις περιορισμούς των βασικών δομών που χρησιμοποιούνταν για να δημιουργήσουν μια «έξυπνη» συμπεριφορά. Συγκεκριμένα μία απλή δομή νευρωνικού δικτύου που είχε δύο εισαγωγές δεδομένων δεν μπορούσε να προγραμματιστεί για να αναγνωρίζει πότε τα δεδομένα των εισαγωγών αυτών ήταν διαφορετικά. Παρόλο που τα αποτελέσματα δεν μπορούσαν να εφαρμοστούν σε πιο πολύπλοκα και πολυεπίπεδα δίκτυα η έρευνα για τα νευρωνικά δίκτυα ήταν μηδαμινή. Αργότερα συνέβη μια ειρωνική κατάσταση κατά την οποία στα τέλη της δεκαετίας του ’80 οι αλγόριθμοι που χρησιμοποιούνταν χρησιμοποιούσαν στη βάση τους αλγόριθμους που ανακαλύφθηκαν το 1969.

Η εικόνα της λύσης των προβλημάτων που δημιουργήθηκε κατά τη διάρκεια της πρώτης δεκαετίας της έρευνας της ΤΝ ήταν για ένα γενικό ερευνητικό μηχανισμό

ο οποίος προσπαθούσε να ενώσει απλά λογικά βήματα για να βρει ολοκληρωμένες λύσεις. Αυτοί οι μηχανισμοί ονομάστηκαν «αδύναμοι» γιατί παρόλο που ήταν γενικοί, δεν μπορούσαν να αναβαθμιστούν και να χρησιμοποιηθούν σε πιο δύσκολα ή μεγαλύτερα προβλήματα. Η εναλλακτική λύση στους αδύναμους αυτούς μηχανισμούς είναι να χρησιμοποιηθεί πολύ πιο δυνατή και συγκεκριμένη γνώση η οποία επιτρέπει μεγαλύτερα λογικά βήματα και μπορεί να χειριστεί πολύ πιο εύκολα προβλήματα που χρειάζονται εξειδίκευση.

Το πρόγραμμα DENTRAL (Buchanan et al., 1969)(24) ήταν ένα πρώιμο παράδειγμα των παραπάνω προσεγγίσεων. Το πρώτο πρόβλημα που κλήθηκε να λύσει ήταν το πρόβλημα της ανάλυσης μιας μοριακής δομής που δίνεται από ένα φασματογράφο μάζας. Η εισαγωγή του προγράμματος αποτελείται από την αρχική φόρμουλα του μορίου (π.χ. $C_6H_{13}NO_2$) και το φάσμα της μάζας που μας δίνει τις μάζες από τα διάφορα μέρη του ατόμου που παράγονται όταν βομβαρδίζεται από μία δέσμη ηλεκτρονίων. Η απλοϊκή έκδοση του προγράμματος υπολόγισε όλες τις πιθανές δομές που είναι συμβατές με τη φόρμουλα και μετά πρόβλεψε τι φάσμα της μάζας θα παρατηρούνταν για κάθε δομή συγκρίνοντάς το με το πραγματικό φάσμα. Όπως είναι επόμενο, αυτό είναι πάρα πολύ δύσκολο να γίνει για μόρια με μεγάλο μέγεθος. Οι δημιουργοί του DENTRAL συμβουλεύτηκαν διάφορους αναλυτικούς χημικούς και βρήκαν ότι μπορούσαν να υπολογίσουν γνωστά μόρια πάνω στις κορυφές του φάσματος και στη συνέχεια μπορούσαν να υπολογίσουν τις απλές υποδομές του μορίου.

Το πρώτο μεγάλο επιτυχές εμπορικό «έξυπνο» σύστημα, το R1, ξεκίνησε να λειτουργεί στην Digital Equipment Corporation. Το πρόγραμμα βοηθούσε στη διαμόρφωση των παραγγελιών για νέους υπολογιστές. Μέχρι το 1986 είχε κερδίσει για την εταιρία περίπου 40 εκατ. δολάρια το χρόνο. Μέχρι το 1988 η ομάδα ΤΝ της DEC είχε 40 «έξυπνα» συστήματα και συνεχώς ανέπτυσσε καινούρια. Η Du Pont είχε 100 σε χρήση και άλλα 500 σε ανάπτυξη κερδίζοντας 10 εκατ. δολάρια το χρόνο. Σχεδόν κάθε μεγάλη αμερικάνικη εταιρία είχε τη δική της ομάδα ΤΝ και είτε χρησιμοποιούσε κάποιο «έξυπνο» σύστημα ή το ανέπτυσσε.

Το 1981 οι Ιάπωνες ανακοίνωσαν το «Fifth Generation Project» το οποίο ήταν ένα πρόγραμμα για 10 χρόνια με το οποίο θα κατασκεύαζαν υπολογιστές με ΤΝ χρησιμοποιώντας την Prolog. Απαντώντας οι ΗΠΑ δημιούργησαν την Microelectronics and Computer Technology Corporation (MCC) σαν ένα ερευνητικό κέντρο σχεδιασμένο να εξασφαλίσει την εθνική ανταγωνιστικότητα. Και στις δύο

περιπτώσεις η ΤΝ αποτέλεσε μία πολύ μεγάλη προσπάθεια περιλαμβάνοντας σχεδιασμούς microchip και έρευνα πάνω στο ανθρώπινο interface. Παρόλα αυτά τα μέρη της ΤΝ, των MCC και Fifth Generation δεν κατάφεραν να φτάσουν στους τελικούς στόχους τους.

Γενικά, η βιομηχανία της ΤΝ εκτοξεύτηκε από μερικά εκατομμύρια δολάρια το 1980 σε δισεκατομμύρια δολάρια το 1988. Σύντομα μετά από αυτό ήρθε η περίοδος που ονομάζεται «ο Χειμώνας της ΤΝ» όπου πολλές εταιρίες υπέφεραν καθώς απέτυχαν να φέρουν εις πέρας τις εξωφρενικές υποσχέσεις τους.

Παρόλο που η επιστήμη των υπολογιστών είχε εγκαταλείψει κατά κύριο λόγο τον τομέα των νευρωνικών δικτύων στα τέλη της δεκαετίας του ’70, η έρευνα συνεχιζόταν σε άλλα πεδία. Φυσικοί όπως ο John Hopfield(25) (1982) χρησιμοποιούσαν τεχνικές από στατιστικούς μηχανισμούς για να αναλύσουν τη χωρητικότητα και τη βελτίωση των δικτύων και συμπεριφέρονταν στα σύνολα των κόμβων σαν σύνολα ατόμων. Ψυχολόγοι συμπεριλαμβανομένων των David Rumelhart (26) και Geoff Hinton(26) συνέχιζαν τη μελέτη των νευρωνικών δικτύων – μοντέλων της μνήμης. Οι αλγόριθμοι που ανακαλύφθηκαν στα μέσα του ’80 στην ουσία είχαν ήδη ανακαλυφθεί το 1969 από τους Bryson και Ho(27). Οι αλγόριθμοι αυτοί εφαρμόστηκαν σε πολλά προβλήματα στην επιστήμη των υπολογιστών και στην ψυχολογία και τα αποτελέσματα του αλγορίθμου έγιναν ευρέως γνωστά από την εργασία Parallel Distributed Processing το 1986 όπου προκάλεσαν πολύ μεγάλο ενθουσιασμό.

Τα τελευταία χρόνια έχει γίνει επανάσταση και στην έρευνα αλλά και στην μεθοδολογία της έρευνας στην ΤΝ. Είναι πολύ εύκολο πλέον να δουλέψει κάποιος πάνω σε ήδη υπάρχουσες θεωρίες παρά να προτείνει καινούριες, να βασιστεί σε αυστηρά θεωρήματα ή σε πολύ ισχυρές εργαστηριακές αποδείξεις αντί να βασιστεί στην διαίσθησή του και τέλος να μπορεί να εφαρμόσει τα προγράμματά του σε πραγματικές συνθήκες παρά να τα εφαρμόζει σε απλά παιχνίδια.

Από την πλευρά της μεθοδολογίας η ΤΝ χρησιμοποιεί καθαρά επιστημονικές μεθόδους. Για να γίνουν αποδεκτές οι μέθοδοι αυτοί, οι υποθέσεις πρέπει να υποβληθούν σε αυστηρά εμπειρικούς πειραματισμούς, και τα αποτελέσματα θα πρέπει να αναλυθούν στατιστικώς για την σπουδαιότητά τους σύμφωνα με τον Cohen(28) το 1995. Μέσω της πολύ μεγάλης διάδοσης του Διαδικτύου και του εύκολου διαμοιρασμού των δεδομένων και του κώδικα είναι πλέον πολύ εύκολο να επαληθευτούν τα αποτελέσματα των πειραμάτων και των στατιστικών δεδομένων.

Ο τομέας της φωνητικής αναγνώρισης δημιουργεί και διευκρινίζει τα πρότυπα για την ΤΝ. Στην δεκαετία του ’70 μια μεγάλη ποικιλία από διαφορετικές αρχιτεκτονικές και προτάσεις πραγματοποιήθηκαν. Πολλές από αυτές ήταν «ad hoc» και εύθραυστες, και παρουσιάστηκαν μόνο σε κάποια ειδικά επιλεγμένα παραδείγματα. Την πρόσφατη περίοδο κάποιες προτάσεις βασισμένες στα «Hidden Markov Models» (HMM) έχουν κυριαρχήσει στον τομέα αυτό. Δύο πτυχές των HMM είναι σχετικά με το αντικείμενο της φωνητικής αναγνώρισης. Καταρχήν βασίζονται σε αυστηρά μαθηματικά θεωρήματα. Δεύτερον παράγονται από μια διαδικασία εφαρμογής τους σε πραγματικά φωνητικά δεδομένα. Αυτό βεβαιώνει ότι η απόδοση θα είναι υψηλή και σε αυστηρά τεστ τα HMM είχαν πολύ καλά αποτελέσματα και με μεγάλες δυνατότητες βελτίωσης. Η τεχνολογία ομιλίας έχει ήδη μπει στην αγορά και μεγάλες επιχειρήσεις την χρησιμοποιούν ήδη στα συστήματα ασφαλείας τους.

Τα νευρωνικά δίκτυα βρίσκουν εφαρμογή σε αυτό τον τομέα. Πολύ μεγάλο μέρος της έρευνας πάνω στα νευρωνικά δίκτυα στη δεκαετία του ’80 έγινε βασισμένη σε μια προσπάθεια να ανακαλυφθεί το τι μπορούσε να γίνει και να γίνει γνωστό πώς τα νευρωνικά δίκτυα διαφέρουν από τις παραδοσιακές τεχνικές. Χρησιμοποιώντας βελτιωμένη μεθοδολογία και θεωρητικές εφαρμογές, τα νευρωνικά δίκτυα έφτασαν σε τέτοιο σημείο όπου μπορούν να συγκριθούν και να ξεπεράσουν τις τεχνικές που εφαρμόζονταν στη στατιστική και την εκμάθηση μηχανών φτάνοντας σε σημείο να μπορούν να χρησιμοποιηθούν σε κάθε εφαρμογή. Αποτέλεσμα αυτών των εξελίξεων είναι η ανάπτυξη της τεχνολογίας data mining η οποία αποτελεί μια νέα βιομηχανία στο χώρο των υπολογιστών.

Την περίοδο αυτή οι ερευνητές ενθαρρυμένοι από την πρόοδο η οποία πραγματοποιήθηκε στις λύσεις των υποπροβλημάτων της Τ.Ν. ξεκίνησαν να προσπαθούν να δημιουργήσουν έναν «ολοκληρωμένο πράκτορα». Οι ερευνητικές προσπάθειες των Allen (15), Newell(16), John Laird και Paul Rosenbloom (29) στο SOAR (1987-1990) είναι οι καλύτερες και οι πιο γνωστές όσον αφορά τη δημιουργία μιας ολοκληρωμένη αρχιτεκτονικής «ολοκληρωμένου πράκτορα». Η αρχιτεκτονική αυτή σαν σκοπό της έχει την κατανόηση του τρόπου που δουλεύουν οι πράκτορες οι οποίοι είναι προσαρτημένοι σε πραγματικά περιβάλλοντα με συνεχείς εισαγωγές αισθητήρων. Ένα από τα πιο σπουδαία περιβάλλοντα για τους «έξυπνους» πράκτορες είναι το internet. Τα συστήματα της Τ.Ν. έχουν γίνει τόσο κοινά στις εφαρμογές που βασίζονται στο διαδίκτυο ώστε η κατάληξη «bot» έχει γίνει μια καθημερινή γλώσσα.

Ακόμη παραπέρα φτάνουν οι τεχνολογίες της Τ.Ν. στις οποίες βασίζονται πολλά εργαλεία του internet, όπως για παράδειγμα μηχανές αναζήτησης, συστήματα ενημέρωσης και συστήματα κατασκευής ιστοσελίδων.

Ένα από τα βασικά σημεία τα οποία πρέπει να προσεχθούν κατά την κατασκευή «ολοκληρωμένων πρακτόρων» είναι η κατανόηση ότι όλοι οι τομείς της Τ.Ν. θα πρέπει να αναδιοργανωθούν όταν προσπαθούμε να συνδυάσουμε τα αποτελέσματά τους. Πιο συγκεκριμένα σήμερα είναι γενικά αποδεκτό ότι τα συστήματα που χρησιμοποιούν αισθητήρες (sonar, αναγνώριση ομιλίας, εικόνα κτλ) δεν μπορούν να μεταδώσουν τέλεια και αξιόπιστη πληροφορία σχετικά με το περιβάλλον. Γι' αυτό και θα πρέπει τα σχεδιαστικά συστήματα να είναι ικανά να χειριστούν αυτές τις «αβέβαιες» πληροφορίες. Μια δεύτερη μεγάλη συνέπεια της προοπτικής των «έξυπνων» πρακτόρων είναι ότι η Τ.Ν. έχει έρθει πάρα πολύ κοντά με άλλα πεδία ερευνών όπως μαθηματικά και οικονομικά στα οποία επίσης μπορούν να χρησιμοποιηθούν «έξυπνοι» πράκτορες.

2.4 Καθορισμός προβλήματος και λύσης

Ένα πρόβλημα μπορεί να χωριστεί γενικά σε τέσσερα μέρη:

Α) Την αρχική κατάσταση στην οποία ξεκινά ο πράκτορας. Για παράδειγμα η αρχική κατάσταση για τον πράκτορα του παραδείγματός μας στην Βεργίνα θα μπορούσε να εκφραστεί σαν In(Βεργίνα).

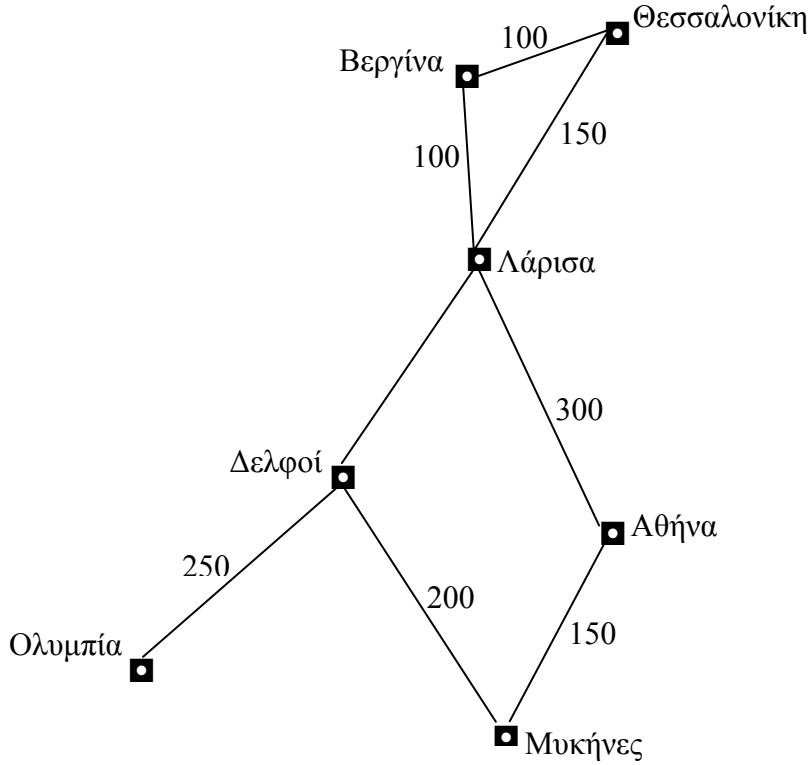
Β) Μια περιγραφή των πιθανών ενεργειών που είναι διαθέσιμες στον πράκτορα. Για το παράδειγμά μας οι επιλογές του πράκτορα είναι δύο και θα μπορούσαν να εκφραστούν ως εξής:

{< Go(Θεσσαλονίκη), In(Θεσσαλονίκη)>, <Go(Λάρισα), In(Λάρισα)>} όπου η κάθε δυάδα <x,y> εκφράζει <Ενέργεια, Νέα_Κατάσταση>.

Μπορούμε να φανταστούμε το πρόβλημα σαν ένα δισδιάστατο σχεδιάγραμμα όπου υπάρχουν οι πόλεις που θέλει να επισκεφθεί ο πράκτορας. Μεταξύ τους συνδέονται με γραμμές δίπλα από τις οποίες αναγράφονται οι αποστάσεις από την μία πόλη στην άλλη. Φυσικά η κίνηση ανάμεσα στις πόλεις είναι αμφίδρομη. Το σχεδιάγραμμα Σχήμα 1 μας δείχνει πώς περίπου μπορούμε να εκφράσουμε το πρόβλημα.

Γ) Η επιβεβαίωση της λύσης, δηλαδή ο συνεχής έλεγχος για το αν κάθε νέο στάδιο στο οποίο βρίσκεται ο πράκτορας είναι και η λύση του προβλήματος. Πολλές φορές βέβαια μπορεί για ένα συγκεκριμένο πρόβλημα να υπάρχουν πολλές λύσεις. Έτσι ο πράκτορας όταν κάνει επιβεβαίωση της λύσης, ελέγχει αν το στάδιο που βρίσκεται τη δεδομένη στιγμή πληροί τις προϋποθέσεις για κάποια από τις πιθανές λύσεις. Στο πρόβλημά μας ο στόχος του πράκτορα είναι να φτάσει σε μία μοναδική κατάσταση και συγκεκριμένα στην Αθήνα. Δηλαδή να ισχύει {In(Αθήνα)}. Πολλές φορές ο στόχος χαρακτηρίζεται από μία συγκεκριμένη κατάσταση στην οποία πρέπει να βρεθεί ο πράκτορας. Για παράδειγμα στο σκάκι για να υπάρξει νικητής θα πρέπει ένας από τους δύο πράκτορες να βρεθεί στην κατάσταση «Ρουά Ματ», στην οποία ο αντίπαλος βασιλιάς δέχεται επίθεση χωρίς να μπορεί να διαφύγει.

Δ) Το κόστος μετακίνησης είναι μια αριθμητική αξία που αντιστοιχεί σε κάθε διαδρομή. Ο πράκτορας που προσπαθεί να επιλύσει το πρόβλημα υπολογίζει το κόστος μετακίνησης από ένα σημείο σε ένα άλλο. Ο τουρίστας που προσπαθεί να φτάσει στην Αθήνα έχει σαν ουσιαστικό του κόστος τον χρόνο που χρειάζεται για να μετακινηθεί από μια πόλη σε μια άλλη. Στην ουσία ο χρόνος είναι η χιλιομετρική απόσταση των πόλεων, αφού παράγοντες στο παράδειγμά μας όπως η ταχύτητα δεν παίζουν ρόλο. Η τελευταία εκφράζεται από τους αριθμούς που υπάρχουν στο σχεδιάγραμμα Σχήμα 1.



Σχήμα 1

Τα ανωτέρω στοιχεία είναι τα κύρια χαρακτηριστικά ενός προβλήματος και μπορούν να συγκεντρωθούν σε μία δομή δεδομένων που εισάγεται σε έναν αλγόριθμο επίλυσης προβλημάτων. Η λύση σε ένα πρόβλημα είναι η όλη διαδρομή που θα κάνει ο πράκτορας από την αρχική κατάσταση μέχρι και την τελική κατάσταση. Το πόσο καλή είναι η λύση και το κατά πόσο είναι τελικά αποδεκτή ελέγχεται από τη συνάρτηση ελέγχου κόστους. Η ιδανικότερη λύση είναι η λύση με τη μικρότερη δυνατή διαδρομή (κόστος) από όλες τις υπόλοιπες λύσεις που έχει υπολογίσει ο πράκτορας.

2.5 Δόμηση και κατανόηση του προβλήματος

Η δόμηση του προβλήματος που έγινε μέχρι στιγμής για το πώς μπορεί να φτάσει ο πράκτορας στην Αθήνα περιλαμβάνει ορισμούς όπως αρχική κατάσταση, έλεγχος τέλους και κόστος κίνησης. Αυτός ο διαχωρισμός μπορεί να φαίνεται επαρκής αλλά αφήνει εκτός υπολογισμού πολλούς παράγοντες του πραγματικού κόσμου. Χαρακτηρίσαμε απλά την μετακίνηση στην Αθήνα σαν In(Αθήνα) ενώ στην πραγματικότητα για να μετακινηθεί κάποιος στην Αθήνα θα συναντήσει στο δρόμο του εμπόδια όπως ελέγχους Τροχαίας, άσχημο καιρό, κάποια αρρώστια, την κατάσταση του δρόμου κλπ. Όλα αυτά βέβαια δεν εγκλείονται στο πρόβλημα γιατί απλά πρέπει να βρεθεί ο γρηγορότερος δρόμος για την Αθήνα. Η διαδικασία εξαγωγής λεπτομερειών από μία αναπαράσταση προβλήματος ονομάζεται αφαίρεση.

Η αφαίρεση των λεπτομερειών γίνεται επάνω στις ενέργειες που κάνει ο πράκτορας κατά τη διάρκεια της μετακίνησής του από μία πόλη σε μία άλλη. Για παράδειγμα ο πράκτορας μπορεί να μετακινηθεί από την Θεσσαλονίκη στη Λάρισα με ήλιο και με ανοικτό το ραδιόφωνο στο αυτοκίνητο χωρίς να τον σταματήσει η Τροχαία ούτε μία φορά και αντίθετα στην διαδρομή Λάρισα – Αθήνα να βρέχει καταρρακτωδώς, χωρίς να ακούει κάποιον σταθμό στο ράδιο, αλλά και να τον σταματήσει τρεις φορές η Τροχαία. Φυσικά σε αυτό το επίπεδο λεπτομερειών δεν περιλαμβάνουμε παράγοντες όπως «στρίψιμο του τιμονιού κατά 30 μοίρες».

Για να μπορέσουμε να προσδιορίσουμε το επίπεδο των λεπτομερειών που θα επιτρέψουμε να επηρεάζουν τον πράκτορα κατά την κίνησή του θα πρέπει να σκεφτούμε το κατά πόσο επιτρέπουν ή απαγορεύουν στον πράκτορα να φτάσει στον προορισμό του. Για παράδειγμα, ας αναλύσουμε μια πιθανή λύση για το πρόβλημά

μας: από Βεργίνα πηγαίνει Θεσσαλονίκη, Λάρισα, Αθήνα. Η λύση αυτή περιλαμβάνει και πιο λεπτομερείς μετακινήσεις όπως το να μετακινηθεί ο πράκτορας από Βεργίνα για Θεσσαλονίκη με ανοικτό το ράδιο και έπειτα να το κλείσει για το υπόλοιπο ταξίδι. Η αφαίρεση είναι αποδεκτή όταν μπορούμε να επεκτείνουμε οποιαδήποτε λύση στην οποία έχουν γίνει αφαιρέσεις λεπτομερειών, σε άλλες λύσεις οι οποίες περιέχουν περισσότερες λεπτομέρειες μεν αλλά χωρίς να αλλάζουν το τελικό αποτέλεσμα. Η αφαίρεση είναι χρήσιμη όταν το να εκτελεστούν όλες οι ενέργειες της λύσης είναι πιο εύκολο από ότι στο αρχικό πρόβλημα. Η επιλογή μιας αποδεκτής και χρήσιμης αφαίρεσης λεπτομερειών περιλαμβάνει την αφαίρεση όσο το δυνατόν περισσότερων λεπτομερειών διατηρώντας όμως την ορθότητα του προβλήματος και της λύσης του.

2.6 Εύκολα Προβλήματα

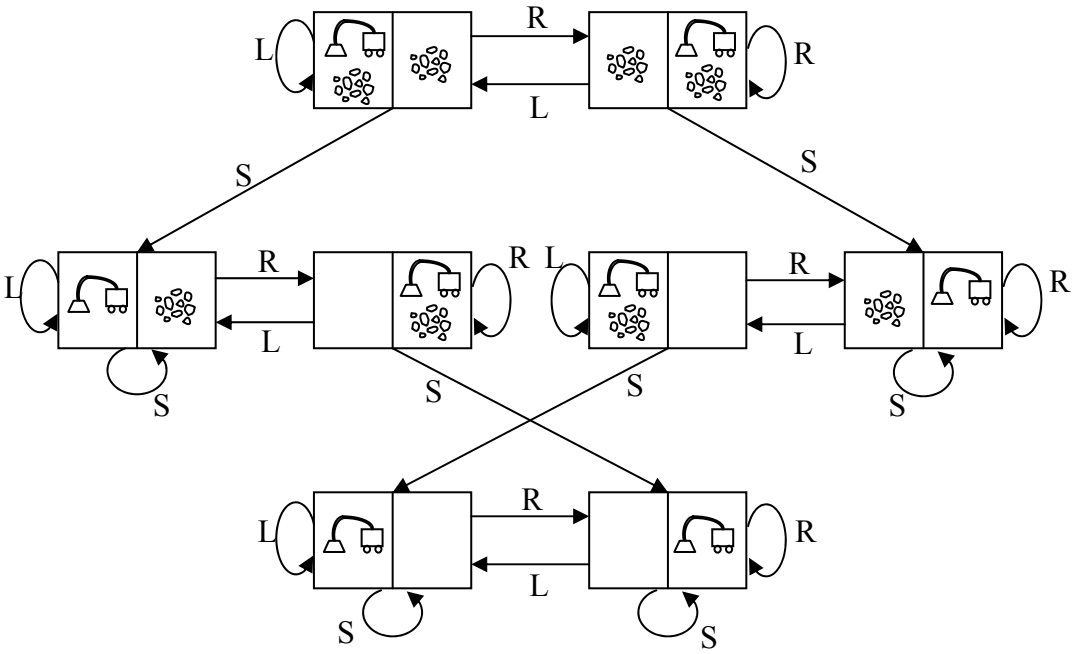
Οι αλγόριθμοι αναζήτησης χρησιμοποιούνται κατά κόρον σε προβλήματα που βρίσκονται σε ένα δεδομένο περιβάλλον. Τα προβλήματα αυτά διαχωρίζονται σε προβλήματα παιγνίων (απλά προβλήματα) και προβλήματα πραγματικού περιβάλλοντος.

2.6.1 Πρόβλημα «ηλεκτρικής σκούπας»

Ένα από τα πιο γνωστά προβλήματα που μπορεί να λυθεί με αλγόριθμο αναζήτησης είναι και το πρόβλημα της «ηλεκτρικής σκούπας» (Artificial Intelligence a Modern Approach, Russel – Norvig, σελ. 65, 2003). Η δομή και ανάλυση αυτού του προβλήματος είναι η παρακάτω:

- **Καταστάσεις:** Ο πράκτορας βρίσκεται σε μία από τις δύο θέσεις, η καθεμία από τις οποίες μπορεί να περιέχει ή όχι σκόνη. Έτσι μπορεί να υπάρχουν $2x2^2 = 8$ πιθανές καταστάσεις.
- **Αρχική Κατάσταση:** Οποιαδήποτε κατάσταση μπορεί να θεωρηθεί σαν αρχική κατάσταση.
- **Συνάρτηση Αναζήτησης:** Εδώ δημιουργούνται οι αποδεκτές καταστάσεις που παράγονται αφού γίνουν οι τρεις βασικές ενέργειες (Αριστερά, Δεξιά, Απορρόφηση). Το ολοκληρωμένο σχήμα φαίνεται στο Σχήμα 2.
- **Επιβεβαίωση της Λύσης:** Εδώ ελέγχεται αν όλα τα τετράγωνα είναι καθαρά.

- **Κόστος Κίνησης:** Κάθε βήμα κοστίζει 1, έτσι το κόστος κίνησης συμπίπτει με τον αριθμό των βημάτων.



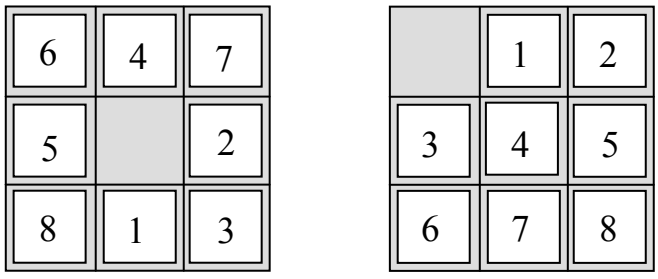
Σχήμα 2

Συγκρινόμενα με τον πραγματικό κόσμο, αυτό το πρόβλημα παιγνίου έχει ορισμένες τοποθεσίες, ορισμένη σκόνη για καθάρισμα, αξιόπιστο καθάρισμα και δεν ξαναπέφτει σκόνη όπου είναι ήδη καθαρισμένο. Ένα σημαντικό σημείο είναι ότι η κάθε κατάσταση χαρακτηρίζεται και από την θέση του πράκτορα αλλά και από τις θέσεις που υπάρχει σκόνη. Γενικά ισχύει ότι οι συνολικές καταστάσεις που μπορεί να υπάρχουν σε ένα τέτοιο πρόβλημα είναι $n2^n$.

2.6.2 Παζλ 8 Θέσεων

Ένα άλλο πρόβλημα είναι το παζλ 8 Θέσεων που αποτελείται από έναν πίνακα 3x3 με οκτώ αριθμημένα τετράγωνα από το 1 έως το 8 και ένα τετράγωνο κενό. Τα τετράγωνα που είναι δίπλα στο κενό μπορούν να κινηθούν προς το κενό. Σκοπός του παιγνίου είναι να τοποθετηθούν οι αριθμοί στη σειρά και όπως φαίνονται στο Σχήμα 3. Η δομή του προβλήματος έχει ως εξής:

- **Καταστάσεις:** Μία περιγραφή της κατάστασης διευκρινίζει την τοποθεσία καθενός από τα οκτώ αριθμημένα κομμάτια και του κενού τετραγώνου στον πίνακα εννέα θέσεων.
- **Αρχική κατάσταση:** Οποιαδήποτε κατάσταση μπορεί να χαρακτηριστεί σαν αρχική.(π.χ σχήμα 3α). Πρέπει να σημειώσουμε ότι οποιοσδήποτε στόχος μπορεί να λυθεί από ακριβώς τις μισές αρχικές καταστάσεις.
- **Συνάρτηση Αναζήτησης:** Εδώ δημιουργούνται οι αποδεκτές καταστάσεις που παράγονται αφού γίνουν οι τέσσερις βασικές κινήσεις (Αριστερά, Δεξιά, Επάνω, Κάτω).
- **Επιβεβαίωση Λύσης:** Ελέγχεται το κατά πόσο η κάθε κατάσταση συμπίπτει με την λύση του σχήματος 3β.
- **Κόστος Κίνησης:** Κάθε βήμα κοστίζει 1, έτσι το κόστος κίνησης συμπίπτει με τον αριθμό των βημάτων.



(α) (β)

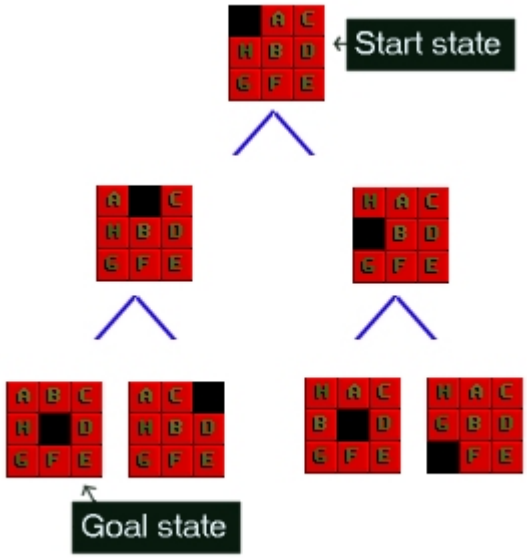
Σχήμα 3

Οι λεπτομέρειες που έχουν αφαιρεθεί από το πρόβλημα περιλαμβάνουν μπλοκάρισμα της κίνησης των τετραγώνων ή ακόμη και την αφαίρεση κάποιων κομματιών με μαχαίρι και την επανατοποθέτησή τους. Στο παράδειγμά μας δεν περιλαμβάνουμε τις ιδιότητες του φυσικού κόσμου.

Το παζλ 8 Θέσεων έχει 9!/2 = 181.440 αποδεκτές καταστάσεις και λύνεται εύκολα. Το 9! είναι το σύνολο των αρχικών καταστάσεων και το διαιρούμε με το δύο, γιατί στην ουσία οι μισές αρχικές καταστάσεις μπορούν να λύσουν οποιοδήποτε στόχο και αν δοθεί. Αυτό συμβαίνει εξαιτίας της συμμετρικότητας του σχεδιασμού του παζλ. Οποιεσδήποτε και αν είναι οι διαστάσεις του μπορούν οι μισές αρχικές καταστάσεις να λύσουν οποιοδήποτε στόχο. Το παζλ 15 Θέσεων (σε πίνακα 4x4) έχει περίπου 1,3 τρισεκατομμύρια καταστάσεις και οι καλύτεροι αλγόριθμοι το λύνουν σε

μερικά milliseconds. Το παζλ 24 Θέσεων (σε πίνακα 5x5) έχει περίπου 10^{25} πιθανές καταστάσεις και θεωρείται δύσκολο στη λύση του.

Μπορούμε να εξετάσουμε τον γρίφο των κεραμιδιών σε ένα 3*3 πλέγμα λεπτομερέστερα. όπου ένα κεραμίδι λείπει και μπορούμε να κινήσουμε τα άλλα κεραμίδια στο κενό έως ότου φτάσουμε στην τελική κατάσταση που είναι και ο στόχος. Παρατηρούμε το Σχήμα 8.



Σχήμα 8:Ο γρίφος των 8–τετραγώνων είναι ένα πολύ απλό παράδειγμα

Υπάρχουν 362.880 διαφορετικές καταστάσεις και για να βρεθεί μια λύση, πρέπει ο πράκτορας να βρει μια διαδρομή μέσα από αυτές. Βέβαια οι μισές από αυτές τις καταστάσεις αρκούν για να βρεθεί οποιαδήποτε λύση. Από τις περισσότερες θέσεις της αναζήτησης ο αριθμός ακρών (οι μπλε γραμμές) είναι δύο. Αυτό σημαίνει ότι ο αριθμός κόμβων που υπάρχουν σε κάθε επίπεδο της αναζήτησης είναι 2^d όπου το d είναι το βάθος. Εάν ο αριθμός βημάτων για να λυθεί μια συγκεκριμένη κατάσταση είναι 18, τότε οι κόμβοι του που θα πρέπει να προσπελαστούν είναι 262.144 σε εκείνο το επίπεδο.

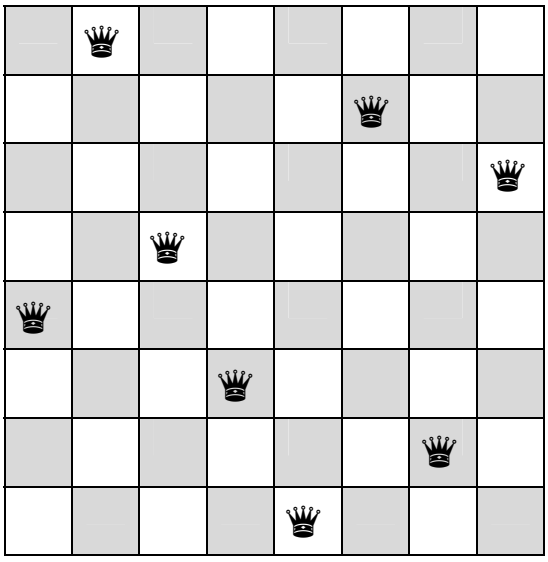
Οι κανόνες που μπορούμε να εφαρμόσουμε στο γρίφο είναι επίσης απλοί. Εάν υπάρχει ένα κενό κεραμίδι επάνω, κάτω, αριστερά ή δεξιά ενός δεδομένου κεραμιδιού, κατόπιν μπορούμε να κινήσουμε εκείνο το κεραμίδι στο διάστημα. Για

να λύσουμε το γρίφο πρέπει να βρούμε την πορεία από την αρχική κατάσταση, μέσω της γραφικής παράστασης στην τελική κατάσταση – στόχο.

2.6.3 ν – Βασίλισσες

Ο στόχος του προβλήματος των 8–Βασιλισσών είναι να τοποθετηθούν 8 βασίλισσες σε μια σκακιέρα με τέτοιο τρόπο ώστε να μην απειλείται καμία. Για να επιλυθεί το πρόβλημα υπάρχουν δύο τρόποι. Ο πρώτος περιλαμβάνει την σκακιέρα άδεια και τοποθετούνται μία – μία οι βασίλισσες. Ο δεύτερος τοποθετεί τυχαία τις βασίλισσες στη σκακιέρα και έπειτα τις κινεί μέχρι να βρει τη λύση. Ακόμη ένα χαρακτηριστικό του προβλήματος είναι η έλλειψη κόστους κίνησης καθώς μετράει μόνο η επίτευξη του στόχου. Η λύση του φαίνεται από το Σχήμα 4.

- **Καταστάσεις:** Οποιαδήποτε τοποθέτηση των 0 έως και 8 βασιλισσών στη σκακιέρα είναι μία αποδεκτή κατάσταση.
- **Αρχική Κατάσταση:** Καμία βασίλισσα στη σκακιέρα.
- **Συνάρτηση Αναζήτησης:** Τοποθέτηση μιας βασίλισσας σε κάποιο άδειο τετράγωνο.
- **Επιβεβαίωση Λύσης:** 8 βασίλισσες στη σκακιέρα και δεν απειλείται καμία.



Σχήμα 4

Σε αυτή τη δομή έχουμε $64 \times 63 \times \cdots \times 57 \approx 3 \times 10^{14}$ πιθανές καταστάσεις για να τις αναλύσουμε. Μία πιο αποτελεσματική ανάλυση του προβλήματος περιλαμβάνει

απαγόρευση τοποθέτησης της βασίλισσας σε κάποιο τετράγωνο που μπορεί να απειληθεί εξαρχής:

- **Καταστάσεις:** Τοποθετήσεις των n βασιλισσών ($0 \le n \le 8$), μία σε κάθε στήλη ξεκινώντας από την άκρα αριστερή, χωρίς να απειλείται καμία, θεωρούνται πιθανές καταστάσεις.
- **Συνάρτηση Αναζήτησης:** Τοποθέτηση μιας βασίλισσας σε κάποιο άδειο τετράγωνο ξεκινώντας από την άκρα αριστερή στήλη, έτσι ώστε να μην απειλείται από καμία άλλη βασίλισσα.

Η παραπάνω ανάλυση μειώνει τις πιθανές καταστάσεις από $64 \times 63 \times \cdots \times 57 \approx 3 \times 10^{14}$ σε 2.057 και πολύ εύκολες λύσεις. Από την άλλη μεριά για 100 βασίλισσες η αρχική ανάλυση του προβλήματος μας δίνει 10^{400} πιθανές καταστάσεις ενώ η βελτιωμένη ανάλυση έχει περίπου 10^{52} καταστάσεις.

2.7 Προβλήματα στον Πραγματικό Κόσμο

Οι αλγόριθμοι αναζήτησης χρησιμοποιούνται σε πολλές εφαρμογές όπως στα δίκτυα υπολογιστών, σε στρατιωτικούς σχεδιασμούς ασκήσεων και σε συστήματα σχεδιασμού αεροπορικών αερογραμμών. Ένα παράδειγμα απλοποιημένο για ένα αεροπορικό ταξίδι είναι το εξής:

- **Καταστάσεις:** Καθεμία αντιπροσωπεύεται από μια τοποθεσία (π.χ. αεροδρόμιο) και ο χρόνος τη δεδομένη στιγμή.
- **Αρχική Κατάσταση:** Διευκρινίζεται από το πρόβλημα.
- **Συνάρτηση Αναζήτησης:** Εδώ περιλαμβάνονται οι προγραμματισμένες πτήσεις, οι καθυστερημένες, καθώς και ο χρόνος πτήσης από το ένα αεροδρόμιο στο άλλο.
- **Επιβεβαίωση Λύσης:** Αν έχει φτάσει η πτήση στον προορισμό της μέσα σε ένα συγκεκριμένο χρονικό διάστημα.
- **Κόστος Κίνησης:** Εξαρτάται από το χρόνο αναμονής, τη διάρκεια της πτήσης, τα τελωνεία και τους ελέγχους των επιβατών, τον τύπο του αεροσκάφους, την ώρα που γίνεται το ταξίδι κτλ.

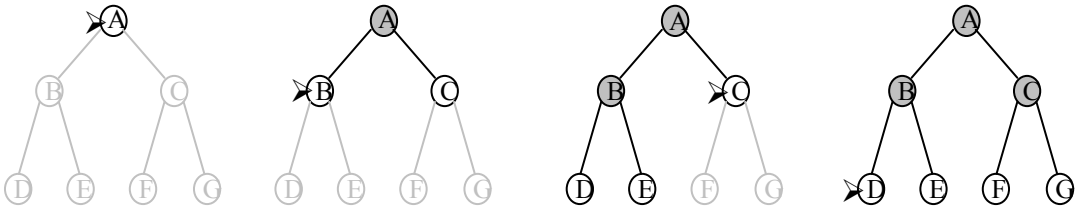
Η πλοήγηση ενός ρομπότ στην ουσία αποτελεί μια γενίκευση των προβλημάτων αναζήτησης. Αντί να χρησιμοποιεί μια δεδομένη διαδρομή, το ρομπότ μπορεί να κινείται ελεύθερα στο χώρο και να κάνει θεωρητικά άπειρες κινήσεις. Όταν ένα ρομπότ κινείται σε μια επίπεδη επιφάνεια τότε κινείται σε δύο διαστάσεις. Αν έχει πόδια, χέρια ή τροχούς που κινούνται ή γενικά μπορεί και κάνει διάφορες ενέργειες, τότε ο χώρος αναζήτησης γίνεται πολυδιάστατος. Στον πραγματικό κόσμο πρέπει να αντιμετωπισθούν και προβλήματα όπως λάθη στο λογισμικό ή και στο hardware του ρομπότ.

Ακόμη, στην σύγχρονη εποχή έχει αυξηθεί η ζήτηση «έξυπνων πρακτόρων» οι οποίοι χρησιμοποιούνται στο διαδίκτυο για αναζήτηση στον ιστό, όπως αναζήτηση απαντήσεων σε προβλήματα και γενικότερα για οποιαδήποτε πληροφορία. Η δομή του διαδικτύου βολεύει για τέτοιου είδους αναζητήσεις, γιατί μπορούμε να δούμε την κάθε μία ιστοσελίδα σαν ξεχωριστό κόμβο για αναζήτηση που συνδέεται με τους υπόλοιπους γύρω του.

2.8 Αναζήτηση κατά Πλάτος

Η αναζήτηση κατά πλάτος είναι μια απλή στρατηγική κατά την οποία ο κόμβος – ρίζα ελέγχεται πρώτος και μετά όλοι οι κόμβοι που ακολουθούν. Στη συνέχεια ελέγχονται οι κόμβοι – παιδιά των δεύτερων κ.ο.κ. Γενικά επεκτείνονται όλοι οι κόμβοι σε ένα δεδομένο βάθος πριν οι κόμβοι από το επόμενο επίπεδο επεκταθούν με τη σειρά τους.

Η αναζήτηση κατά πλάτος μπορεί να χαρακτηρισθεί σαν μια FIFO (First In First Out) αναζήτηση. Σε αυτό το είδος αναζήτησης οι κόμβοι οι οποίοι επισκέπτονται πρώτοι από τον πράκτορα, επεκτείνονται και πρώτοι. Το Σχήμα 5 μας δείχνει τον τρόπο που επεκτείνονται οι κόμβοι στην αναζήτηση κατά πλάτος.



Σχήμα 5

Όσον αφορά την αναζήτηση κατά πλάτος μπορούμε να θέσουμε μερικούς χαρακτηρισμούς. Καταρχήν είναι ολοκληρωμένη, δηλαδή έστω και αργά θα βρει τη λύση. Αυτό συμβαίνει γιατί ακόμη και αν ο κόμβος – λύση είναι στο μέγιστο βάθος, ο πράκτορας θα τον βρει γιατί αφού επεκτείνει όλους τους κόμβους όλων των επιπέδων εκτός του τελευταίου θα επεκτείνει και το τελευταίο επίπεδο και θα βρει τη λύση. Βέβαια αν βρεθεί κάποιος κόμβος – λύση πριν επεκταθούν όλοι οι κόμβοι, δεν σημαίνει ότι θα είναι και η βέλτιστη λύση. Γενικά μπορούμε να πούμε ότι οι αλγόριθμοι που χρησιμοποιούν αναζήτηση κατά πλάτος μπορούν να βρίσκουν τη βέλτιστη λύση όταν το κόστος είναι ίδιο για οποιαδήποτε ενέργεια εκτελέσει ο πράκτορας.

Οι δύο κυριότεροι λόγοι για να μην αποτελεί η αναζήτηση κατά πλάτος την μοναδική μας επιλογή είναι η μνήμη που καταλαμβάνει καθώς και ο χρόνος που χρειάζεται για να ολοκληρωθεί μια αναζήτηση. Για να κατανοήσουμε επακριβώς τους δύο αυτούς παράγοντες θα κάνουμε έναν υπολογισμό. Ας υποθέσουμε ότι έχουμε ένα πρόβλημα το οποίο για κάθε κόμβο έχει ν προεκτάσεις. Έτσι ξεκινώντας από τη ρίζα της δενδρικής δομής των κόμβων, επεκτείνονται ν κόμβοι και ταυτόχρονα για το αμέσως επόμενο ν^2. Ο κάθε κόμβος έχει ν προεκτάσεις οπότε το επόμενο επίπεδο μας δίνει ν^3. Ας υποθέσουμε ότι ο κόμβος – λύση βρίσκεται σε βάθος ίσο με α. Στη χειρότερη περίπτωση θα επεκταθούν όλοι οι κόμβοι εκτός από τον τελευταίο στο επίπεδο ν, επεκτείνοντας έτσι $\nu^{\alpha+1} - \nu$ κόμβους στο επίπεδο α + 1. Τότε ο συνολικός αριθμός των κόμβων που παράχθηκαν είναι:

$$\nu + \nu^2 + \nu^3 + ... + \nu^{\alpha} + (\nu^{\alpha+1} - \nu) = \mathrm{O}(\nu^{\alpha+1})$$

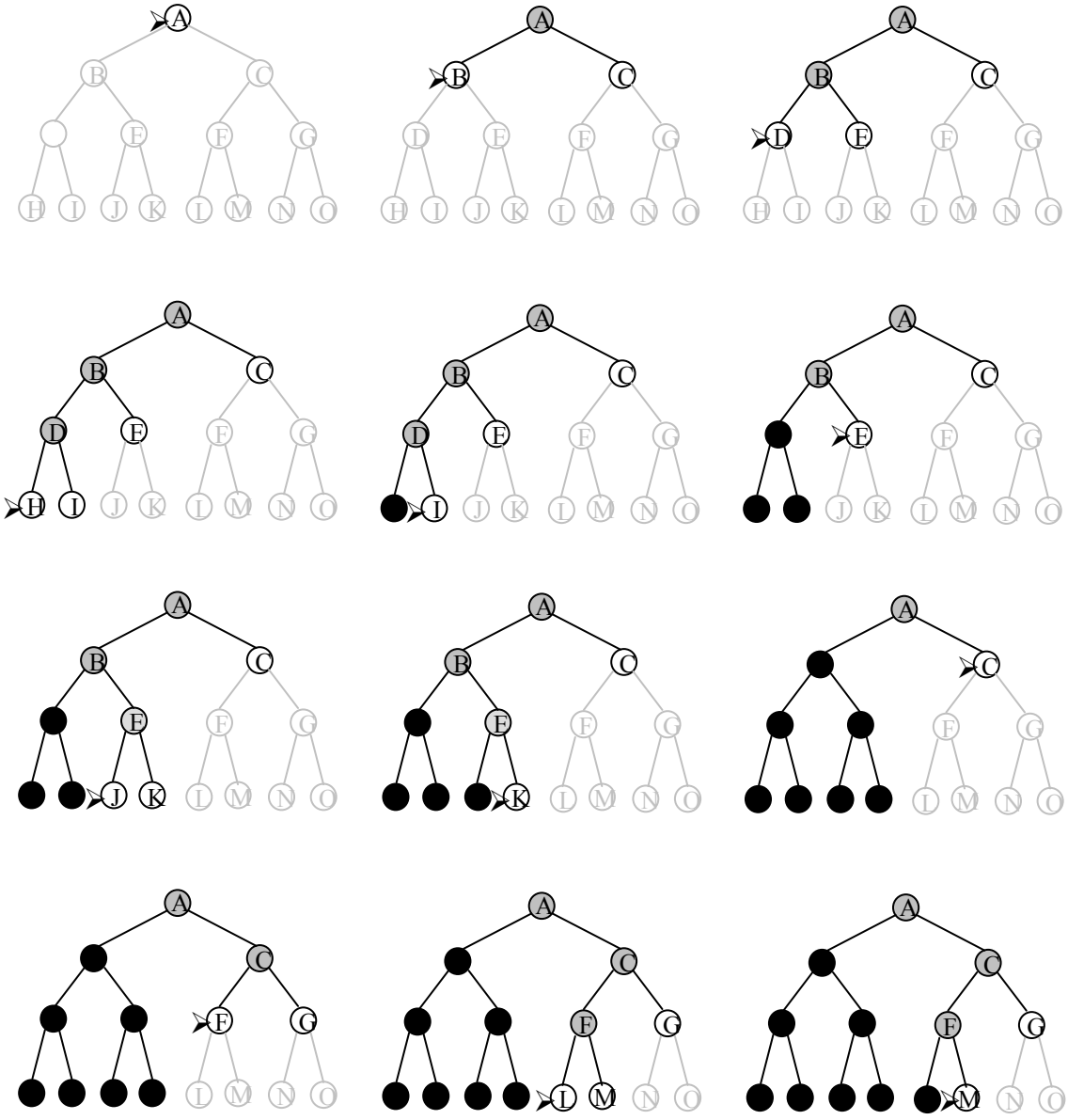
Κάθε κόμβος που επεκτείνεται ή προσπελαύνεται πρέπει να παραμείνει στη μνήμη, γιατί είναι είτε κόμβος – παιδί είτε αρχικός κόμβος. Στο Σχήμα 6 φαίνονται οι απαιτήσεις μιας αναζήτησης κατά πλάτος για ποικίλα βάθη. Το συμπέρασμα που μπορούμε να βγάλουμε είναι ότι ο συγκεκριμένος τρόπος αναζήτησης είναι και χρονοβόρος αλλά πολύ περισσότερο μας κοστίζει σε μνήμη. Με μια γρήγορη παρατήρηση μπορούμε να δούμε ότι σε βάθος 12 για παράδειγμα η λύση θα μας πάρει 35 χρόνια για να βρεθεί και θα δεσμεύσει μνήμη 10 petabytes.

Βάθος	Κόμβοι	Χρόνος	Μνήμη
2	1100	.11 δευτερόλεπτα	1 megabyte
4	111,100	11 δευτερόλεπτα	106 megabytes
6	10^7	19 λεπτά	10 gigabytes
8	10^9	31 ώρες	1 terabyte
10	10^{11}	129 ημέρες	101 terabytes
12	10^{13}	35 χρόνια	10 petabytes
14	10^{15}	3,523 χρόνια	1 exabyte

Σχήμα 6 (Πηγή:Artificial Intelligence a Modern Approach, Russel – Norvig, 2003)

2.9 Αναζήτηση κατά Βάθος

Η αναζήτηση κατά βάθος πάντα επεκτείνει τον βαθύτερο κόμβο που υπάρχει στη σειρά των κόμβων. Επεκτείνει δηλαδή όλους τους κόμβους μέχρι και σε αυτό το βάθος όπου δεν θα υπάρχει άλλος κόμβος προς επέκταση. Χαρακτηριστικό του συγκεκριμένου τρόπου αναζήτησης είναι ότι ο τελευταίος κόμβος είναι και αυτός που ελέγχεται. Δηλαδή ισχύει ο χαρακτηρισμός LIFO (Last In First Out). Ένα παράδειγμα αναζήτησης σε βάθος φαίνεται στο Σχήμα 7.



Σχήμα 7

Η αναζήτηση κατά βάθος έχει πάρα πολύ μικρές απαιτήσεις σε μνήμη. Χρειάζεται μόνο να αποθηκεύσει ένα «μονοπάτι» από τη ρίζα μέχρι και τον κόμβο – κλαδί, καθώς και τους υπόλοιπους κόμβους που δεν έχουν επεκταθεί ακόμη και βρίσκονται στον ίδιο κόμβο – κλαδί. Όταν ένας κόμβος επεκταθεί, τότε μπορεί να αφαιρεθεί από την μνήμη, μόλις επεκταθούν και όλοι οι κόμβοι – παιδιά που βρίσκονται στα επόμενα επίπεδα. Για ένα πρόβλημα που η δενδρική δομή του έχει ν κλαδιά και μέγιστο βάθος μ, τότε η μνήμη που θα δεσμευθεί είναι μόνο νμ + 1 κόμβοι. Υποθέτοντας ότι ο κόμβος λύση δεν έχει άλλα παρακλάδια μπορούμε να

δούμε ότι η αναζήτηση κατά βάθος δέσμευσε 118 kilobytes αντί για 10 petabytes για βάθος α = 12, το οποίο σημαίνει 10 δισεκατομμύρια φορές λιγότερη μνήμη.

2.10 Αναζήτηση σε ορισμένο βάθος

Η αναζήτηση σε ορισμένο βάθος στην ουσία αποτελεί ένα συνδυασμό των αναζητήσεων κατά πλάτος και κατά βάθος. Η λειτουργία του είναι ίδια με την αναζήτηση σε βάθος, με την διαφορά ότι γίνεται μέχρι ένα ορισμένο επίπεδο. Αν δεν βρεθεί λύση στο συγκεκριμένο βάθος, τότε τίθεται ένα νέο βάθος για να ερευνηθεί και συνεχίζεται η αναζήτηση έως ότου βρεθεί η λύση.

2.11 Αναζήτηση βέλτιστης επέκτασης

Εδώ η αναζήτηση που πραγματοποιείται έχει σαν σκοπό της να βρει ποιος κόμβος έχει το μικρότερο κόστος και είναι πιο κοντά στη λύση και να τον επεκτείνει. Ο συγκεκριμένος αλγόριθμος αξιολογεί και εντοπίζει ποιον κόμβο θα επεκτείνει σύμφωνα με μια συνάρτηση αξιολόγησης κόστους. Η συνάρτηση δημιουργεί μια δομή δεδομένων που καταχωρεί τους υποψήφιους προς επέκταση κόμβους, βρίσκει αυτόν με το μικρότερο κόστος και τον επεκτείνει. Για την αναζήτηση της βέλτιστης επέκτασης υπάρχουν και άλλες συναρτήσεις. Μία από τις πιο γνωστές είναι η ευρετική συνάρτηση, η οποία εκφράζεται ως εξής:

h(n) = ελάχιστο κόστος από τον κόμβο n έως τον κόμβο – λύση.

2.12 Αναζήτηση με τον αλγόριθμο A*

Εδώ εξετάζουμε το πρόβλημα της μικρότερης πορείας μεταξύ δύο σημείων σε ένα άγνωστο πραγματικό φυσικό περιβάλλον, στο οποίο ένας κινητός πράκτορας πρέπει να ταξιδέψει στο γύρω του περιβάλλον για να εξερευνήσει τα άγνωστα εδάφη. Η συγκεκριμένη αναζήτηση μπορεί να χαρακτηρισθεί από μια πορεία αναζήτησης η οποία αντιπροσωπεύεται συνήθως με τη μορφή γραφικών παραστάσεων, στις οποίες οι κόμβοι ενώνονται μεταξύ

τους και συνδέονται αμφίδρομα σχηματίζοντας γραφικές παραστάσεις. Η μετακίνηση μεταξύ των κόμβων ορίζεται από τις άκρες γραφικών παραστάσεων. Οι γραφικές παραστάσεις μπορούν να αντιπροσωπεύσουν τα διαφορετικά περιβάλλοντα, όπως οδικούς χάρτες, παίγνια και επικοινωνία δικτύων. Η κίνηση από έναν κόμβο της γραφικής παράστασης προς άλλο μπορεί να γίνει με το να ισχύσει η λογική των πρακτόρων που χειρίζονται την επικρατούσα κατάσταση σε μια δεδομένη στιγμή ή από την κίνηση ενός πραγματικού πράκτορα σε ένα σταθερό περιβάλλον από έναν κόμβο σε κάποιον άλλο. Ο γρίφος του παιγνίου των κεραμιδιών και ο κύβος Rubik είναι παραδείγματα πρώτου τύπου καθώς το περιβάλλον αλλάζει και η αρχική τους κατάσταση είναι τυχαία, ενώ ένας οδικός χάρτης είναι ένα παράδειγμα του δεύτερου τύπου καθώς παραμένει σταθερός και κινείται μόνο ο πράκτορας.

Γραφικές παραστάσεις στα προβλήματα αναζήτησης μπορούν να χωρισθούν στις τρεις ακόλουθες κατηγορίες:

- **Πλήρως γνωστές γραφικές παραστάσεις:** Εάν όλοι οι κόμβοι και οι άκρες μιας γραφικής παράστασης αποθηκεύονται στη μνήμη, η γραφική παράσταση είναι πλήρως γνωστή. Η εισαγωγή για τέτοια προβλήματα είναι συνήθως πλήρης γραφική παράσταση που αντιπροσωπεύεται από μια μήτρα γειτνίασης ή έναν κατάλογο γειτνίασης. Ένα σχετικό πρόβλημα σε αυτήν την περίπτωση θα ήταν να βρεθεί, παραδείγματος χάριν, η κοντύτερη πορεία στο α σημείο του οδικού χάρτη στον οποίο όλοι οι κόμβοι και οι άκρες είναι γνωστοί εκ των προτέρων.
- **Πολύ μεγάλες γραφικές παραστάσεις:** Γραφικές παραστάσεις που λόγω των περιορισμών αποθήκευσης και χρόνου δεν είναι πλήρως γνωστές και δεν μπορούν να αποθηκευτούν πλήρως σε οποιαδήποτε συσκευή αποθήκευσης. Πολλές γραφικές παραστάσεις για τα προβλήματα αναζήτησης έχουν έναν εκθετικό αριθμό κόμβων. Παραδείγματος χάριν, το πρόβλημα 24-κεραμίδι puzzle έχει 10^{25} καταστάσεις και δεν μπορεί να αποθηκευτεί εντελώς στις τρέχουσες μηχανές. Η εισαγωγή για τέτοια προβλήματα διευκρινίζεται συνήθως από τη γενική δομή μιας κατάστασης, το περιβάλλον αναζήτησης, τους διαφορετικούς πράκτορες, μια αρχική κατάσταση, και ένα σύνολο καταστάσεων που αποτελούν τον στόχο.

- **Μικρές, μερικώς γνωστές γραφικές παραστάσεις:** Η τρίτη κατηγορία περιέχει τις γραφικές παραστάσεις που αντιπροσωπεύουν ένα μερικώς γνωστό φυσικό περιβάλλον. Παραδείγματος χάριν, ένας κινητός πράκτορας σε μια άγνωστη περιοχή χωρίς έναν χάρτη δεν έχει την πλήρη γνώση του περιβάλλοντος. Λαμβάνοντας υπόψη το πέρας του χρόνου, πραγματικά ο πράκτορας μπορεί πλήρως να ερευνήσει το περιβάλλον δεδομένου ότι δεν είναι πολύ μεγάλος. Λόγω της μερικής γνώσης, μόνο μια μικρή μερίδα της γραφικής παράστασης δίνεται ως εισαγωγή.

Για την κατηγορία πλήρως γνωστών γραφικών παραστάσεων, κλασσικοί αλγόριθμοι, μπορούν να χρησιμοποιηθούν για να βρεθεί η βέλτιστη πορεία μεταξύ οποιωνδήποτε δύο κόμβων. Αυτοί οι αλγόριθμοι υποθέτουν ότι κάθε κόμβος της γραφικής παράστασης μπορεί να προσεγγιστεί από τον αλγόριθμο σε σταθερό χρόνο. Αυτή η υπόθεση ισχύει δεδομένου ότι όλοι οι κόμβοι και οι άκρες της γραφικής παράστασης είναι γνωστοί εκ των προτέρων και αποθηκεύονται μέσα στη μνήμη του υπολογιστή. Κατά συνέπεια η χρονική πολυπλοκότητα αυτών των αλγορίθμων μετριέται από τον αριθμό κόμβων και άκρων που επεξεργάζονται κατά τη διάρκεια της αναζήτησης.

Για τη δεύτερη κατηγορία γραφικών παραστάσεων οι βασικοί αλγόριθμοι δεν είναι συνήθως αποδοτικοί, επειδή ο αριθμός κόμβων στη γραφική παράσταση είναι πολύ μεγάλος (συνήθως εκθετικό). Επίσης, μόνο μια πολύ μικρή μερίδα της γραφικής παράστασης αποθηκεύεται στη μνήμη οποιαδήποτε στιγμή. Ο αλγόριθμος A* και οι γραμμικές εκδόσεις του, π.χ. IDAκαι RBFS, είναι οι κοινές μέθοδοι για τις κοντύτερες πορείες στις μεγάλες γραφικές παραστάσεις. Ο A* κρατά ανοικτό κατάλογο των κόμβων που έχουν παραχθεί αλλά δεν έχουν επεκταθεί ακόμα, και επιλέγεται ο καλύτερος κόμβος για επέκταση. Όταν ένας κόμβος επεκτείνεται, κινείται από τον ανοικτό κατάλογο στον κλειστό κατάλογο και οι γειτονικοί του κόμβοι που παράγονται τίθενται στον κατάλογο προς έλεγχο. Η αναζήτηση ολοκληρώνεται όταν επιλέγεται ένας κόμβος στόχου για την επέκταση ή όταν ο ανοικτός κατάλογος είναι κενός. Η συνάρτηση κόστους του A* είναι $\varphi(v) = \gamma(v) + h(v)$, όπου το $\gamma(v)$ είναι η απόσταση που διανύεται από την αρχική κατάσταση μέχρι το ν και το $h(v)$ μια ευρετική εκτίμηση του κόστους από τον κόμβο *η* στο στόχο. Εάν το $h(v)$ δεν υπερβαίνει ποτέ το πραγματικό κόστος από τον κόμβο *η* στο στόχο, λέμε ότι το $h(v)$ είναι αποδεκτό. Κατά τη χρησιμοποίηση ενός

αποδεκτού ευρετικού *h(v)* , ο A* αποδείχθηκε αποδεκτός, πλήρης, και βέλτιστα αποτελεσματικός. Με άλλα λόγια, ένας τέτοιος ευρετικός αλγόριθμος όπως ο A* είναι εγγυημένος για να επιστρέψει πάντα την κοντύτερη πορεία. Επιπλέον, οποιοσδήποτε άλλος αλγόριθμος που υποστηρίζει ότι επιστρέφει τη βέλτιστη πορεία πρέπει να επεκτείνει τουλάχιστον όλους τους κόμβους που επεκτείνει ο A * δεδομένου ότι έχουν τον ίδιο κόμβο – στόχο προς εύρεση.

Ένας κύκλος επέκτασης κόμβων από τον A* πραγματοποιείται σε σταθερό χρόνο. Αυτό είναι επειδή παίρνει ένα σταθερό χρονικό διάστημα για να ανακτηθεί ένας κόμβος από τον ανοικτό κατάλογο και να παραχθούν όλοι οι γείτονές του. Κατά συνέπεια η πολυπλοκότητα του A* μπορεί επίσης να μετρηθεί από τον αριθμό των παραγόμενων κόμβων. Εδώ εξετάζουμε την εύρεση της κοντύτερης πορείας στις γραφικές παραστάσεις της τρίτης κατηγορίας, δηλαδή τις μικρές, μερικώς γνωστές γραφικές παραστάσεις που αντιστοιχούν σε ένα πραγματικό φυσικό περιβάλλον. Αντίθετα από γραφικές παραστάσεις των άλλων δύο κατηγοριών, όπου ένας σταθερός αριθμός υπολογισμών γίνεται για την επέκταση κόμβων, δεν μπορούμε να υποθέσουμε, για αυτόν τον τύπο γραφικών παραστάσεων ότι υπάρχει σταθερός χρόνος υπολογισμού και προσπέλασης των κόμβων. Πολλοί από τους κόμβους και τις άκρες αυτής της γραφικής παράστασης δεν είναι γνωστοί εκ των προτέρων. Επομένως, για να επεκτείνει έναν κόμβο που δεν είναι γνωστός εκ των προτέρων, ένας κινητός πράκτορας πρέπει πρώτα να ταξιδέψει σε εκείνο τον κόμβο προκειμένου να τον εξερευνήσει και να μάθει για τους γείτονές του. Το κόστος της αναζήτησης σε αυτή την περίπτωση είναι το κόστος που έχει ένας πράκτορας που κινείται σε ένα φυσικό περιβάλλον, δηλαδή είναι ανάλογο με την απόσταση που διανύεται από τον πράκτορα. Ένας αποδοτικός αλγόριθμος επομένως θα ελαχιστοποιούσε την απόσταση που διανύεται από τον πράκτορα, έτσι ώστε να βρεθεί η βέλτιστη πορεία. Σημειώστε ότι αφού οι μικρές γραφικές παραστάσεις εξετάζονται εδώ, μπορούμε να παραλείψουμε τον πραγματικό χρόνο υπολογισμού και να εστιάσουμε μόνο στο ταξίδι του πράκτορα. Εδώ προτείνεται ο αλγόριθμος A(PHA*)(6) (Journal of Artificial Intelligence Research 21 (2004)) για την επίλυση αυτού του είδους προβλημάτων. Ο PHA* (6)(Journal of Artificial Intelligence Research 21 (2004))επεκτείνει όλους τους υποχρεωτικούς κόμβους που και ο A* θα επέκτεινε και επιστρέφει την κοντύτερη πορεία μεταξύ των δύο σημείων. Εντούτοις, η πολυπλοκότητα του αλγορίθμου

είναι μετρημένη από το κόστος κίνησης του κινούμενου πράκτορα. Όπως περιγράφεται παρακάτω, πολλές φορές ο πράκτορας δεν επιλέγει στην πρώτη κίνηση τους κοντινούς κόμβους αμέσως αλλά ενεργεί σύμφωνα με τις κινήσεις που οριοθετούνται από τον A*. Αντίθετα από τους συνηθισμένους στόχους που τίθενται από την περιήγηση, ο σκοπός του πράκτορα δεν είναι να επιτύχει τον κόμβο – στόχο το συντομότερο δυνατόν, αλλά εξερευνά τη γραφική παράσταση με έναν τέτοιο τρόπο που η κοντύτερη πορεία θα ανακτηθεί για μετέπειτα χρήση.

Παρακάτω είναι δύο παραδείγματα των πραγματικών εφαρμογών που χρησιμοποιούν τον αλγόριθμο A* και τις εξελίξεις του:

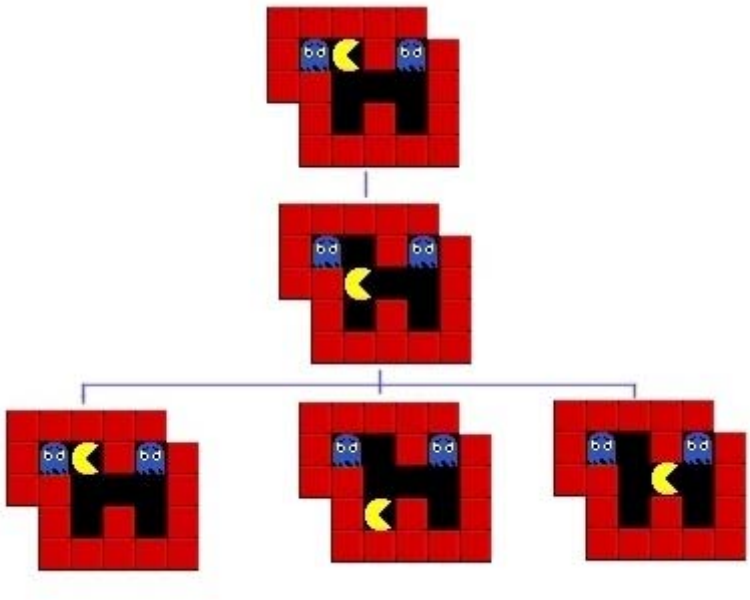
Παράδειγμα 1: Ένα τμήμα των στρατευμάτων διατάζεται να φθάσει σε μια συγκεκριμένη θέση. Οι συντεταγμένες της θέσης είναι γνωστές. Πλοήγηση ολόκληρου του τμήματος μέσα από το άγνωστο εχθρικό έδαφος μέχρι την επίτευξη του προορισμού του είναι αδικαιολόγητα ανεπαρκής όπως φυσικά και αναποτελεσματική καθώς το στράτευμα θα εξοντωθεί. Είναι συνηθισμένο σε αυτή την περίπτωση για μια ομάδα αναζήτησης και ανίχνευσης της καλύτερης πορείας για να μπορέσει ο στρατός να περάσει χωρίς να συναντήσει τους εχθρούς. Οι ανιχνευτές εξερευνούν την έκταση και εκθέτουν την καλύτερη πορεία για να κινηθεί το τμήμα προκειμένου να επιτευχθεί ο προορισμός του κατά τρόπο αποδοτικό.

Χρησιμοποιώντας τον A*, εάν ο ανοικτός κατάλογος κρατά στη μνήμη του μια ως σειρά αναμονής και προτεραιότητας, θα μπορούσαμε να υπολογίσουμε το χρόνο ανάκτησης του καλύτερου κόμβου. Εντούτοις, για πολλά προβλήματα, όπως τα puzzle−κεραμιδιών και ο κύβος Rubik, ένας απλός αλγόριθμος τηρώντας σειρά αναμονής του τύπου First In First Out αρκεί για την επίλυση του γρίφου σύμφωνα με τους Taylor & Korf, (30)(1993). Επιπλέον, για γραμμικές εκδόσεις των αλγορίθμων, όπως οι IDA * και RBFS (που είναι βασισμένοι στην αναζήτηση κατά βάθος), υπάρχει σταθερός χρόνος προσπέλασης κάθε κόμβου. Επίσης υποθέτουμε ότι ο αριθμός των γειτονικών κόμβων είναι πεπερασμένος.

Παράδειγμα 2: Για τους ηλεκτρονικούς υπολογιστές που συνδέονται κατευθείαν με κάποιο δίκτυο ή σε μη απευθείας σύνδεση, ο ρυθμός μετάδοσής τους μπορεί να υποβιβαστεί σοβαρά λόγω υπερφόρτωσης του δικτύου. Επομένως, πολλά δίκτυα δεν μπορούν να αντιπροσωπευθούν με κάποιο σταθερό κόστος κίνησης από κάποιον κόμβο σε έναν άλλο, έτσι δεν μπορούν να σχηματιστούν

πλήρως γνωστές γραφικές παραστάσεις. Μεταφορά των μεγάλων αρχείων (π.χ. αρχεία πολυμέσων) μεταξύ δύο υπολογιστών σε ένα δίκτυο μπορούν συχνά να έχουν μεγάλη χρονική κατανάλωση, μια και τα δεδομένα μπορούν να καθοδηγούνται μέσα από πολλά κανάλια επικοινωνίας και ηλεκτρονικούς υπολογιστές πριν φθάσουν στον προορισμό τους. Η εύρεση της βέλτιστης πορείας μεταξύ αυτών των συγκροτημάτων ηλεκτρονικών υπολογιστών θα μπορούσε να βελτιώσει το χρόνο μεταφοράς των μεγάλων αρχείων. Δεδομένου ότι το δίκτυο δεν μπορεί να μαθευτεί πλήρως, η εύρεση μιας βέλτιστης πορείας μεταξύ δύο κόμβων απαιτεί κάποια εξερεύνηση του δικτύου. Μια αποδοτική και κομψή λύση είναι να σταλούν τα μικρά πακέτα (που λειτουργούν ως ανιχνεύσεις) για να εξερευνήσουν το δίκτυο και να επιστρέψουν τη βέλτιστη πορεία, δεδομένου ότι το δίκτυο είναι σταθερό για τουλάχιστον μια μικρή χρονική περίοδο. Υποθέτοντας ότι ένα συγκρότημα ηλεκτρονικών υπολογιστών στο δίκτυο αναγνωρίζεται μόνο από τα γειτονικά συστήματά του, βρισκόμαστε αντιμέτωποι με το πρόβλημα της βέλτιστης πορείας σε ένα πραγματικό φυσικό περιβάλλον.

Παράδειγμα 3 (Pathfinding): Σε ένα παιχνίδι υπολογιστή, ή κάποιο άλλο pathfinding σενάριο, θέλουμε να βρούμε μια συγκεκριμένη κατάσταση για να μπορέσουμε να φύγουμε και να μεταφερθούμε σε κάποιο άλλο σημείο , χωρίς όμως να συναντηθούμε με κάποιο αντικείμενο ή να υπάρχει πρόσκρουση στους τοίχους ή μετάβαση πάρα πολύ μακριά. Ο αλγόριθμος A* όχι μόνο θα βρει μια πορεία, εάν υπάρχει μια αποδεκτή, αλλά θα βρει και την κοντύτερη πορεία. Μια κατάσταση στο παιχνίδι είναι απλά μια θέση στον κόσμο του παιχνιδιού. Στο παράδειγμα ενός παιχνιδιού λαβυρίνθου, όπως το γνωστό παιχνίδι Pacman, μπορούμε να δούμε ότι το παιχνίδι βασίζεται σε ένα πλέγμα. Η αρχική κατάσταση για ένα φάντασμα, θα ήταν η συντεταγμένη όπου θα βρισκόταν το φάντασμα κατά την έναρξη της αναζήτησης. Η τελική κατάσταση θα ήταν όπου ο pacman θα βρισκόταν έτσι ώστε να μπορεί να «φάει» το φάντασμα.



Σχήμα 9: Τα πρώτα τρία βήματα ενός pathfinding σεναρίου

Είμαστε τώρα έτοιμοι να εξετάσουμε τη λειτουργία του αλγόριθμου A*. Αυτό που πρέπει να κάνουμε είναι να ξεκινήσουμε από την κατάσταση – στόχο και έπειτα να δημιουργήσουμε τη γραφική παράσταση. Ας πάρουμε τον γρίφο των 8 τετραγώνων στο σχήμα 8. Ρωτάμε πόσες κινήσεις μπορούν να γίνουν από την αρχική κατάσταση. Η απάντηση είναι 2. Υπάρχουν δύο κατευθύνσεις που μπορούμε να κινήσουμε το κενό κεραμίδι, και έτσι η γραφική παράστασή μας επεκτείνεται. Εάν επρόκειτο ακριβώς να συνεχίσουμε τυφλά να επεκτείνουμε τους διαδόχους σε κάθε κόμβο, θα μπορούσαμε ενδεχομένως να γεμίσουμε τη μνήμη του υπολογιστή προτού να βρούμε τον κόμβο στόχου. Προφανώς πρέπει να έχουμε αποθηκεύσει τους καλύτερους κόμβους και να ψάξουμε εκείνο τον πρώτο. Πρέπει επίσης να έχουμε αποθηκεύσει τους κόμβους που έχουμε επεκτείνει ήδη, έτσι ώστε δεν θα βρισκόμαστε επανειλημμένα στην ίδια κατάσταση. Αρχίζουμε με τον ΑΝΟΙΚΤΟ κατάλογο. Εδώ είναι όπου θα αποθηκεύσουμε τους κόμβους που δεν έχουμε επεκτείνει ακόμα. Όταν ο αλγόριθμος αρχίζει την επίλυση, η αρχική κατάσταση τοποθετείται στον ανοικτό κατάλογο, γιατί είναι η μοναδική κατάσταση που την γνωρίζουμε και δεν έχουν επεκταθεί περαιτέρω κόμβοι. Έτσι θα επεκτείνουμε τους κόμβους από την κατάσταση έναρξης και θα τοποθετηθούν στον ΑΝΟΙΚΤΟ κατάλογο. Τώρα τελειώσαμε με τον κόμβο έναρξης και θα τον βάλουμε στον ΚΛΕΙΣΤΟ κατάλογο. Ο ΚΛΕΙΣΤΟΣ κατάλογος είναι ένας κατάλογος κόμβων που έχουμε επεκτείνει.

Η χρησιμοποίηση του ΑΝΟΙΚΤΟΥ και ΚΛΕΙΣΤΟΥ καταλόγου μας αφήνει να είμαστε εκλεκτικότεροι για αυτό που θα εξετάσουμε στην συνέχεια της αναζήτησής μας. Θέλουμε να εξετάσουμε τους καλύτερους κόμβους πρώτα. Θα δώσουμε σε κάθε κόμβο ένα αποτέλεσμα για να μπορέσουμε να υπολογίσουμε τον καλύτερο τρόπο για να κινηθεί ο πράκτορας. Αυτό το αποτέλεσμα πρέπει να θεωρηθεί ως κόστος από τον κόμβο στο στόχο συν το κόστος όπου είμαστε, δηλαδή $\varphi = \gamma + h$.

Παραδοσιακά αυτό έχει αντιπροσωπευθεί από τα γράμματα φ, γ και h. Το "γ" είναι το σύνολο του κόστους που υπάρχει για να φτάσει ο πράκτορας εδώ, το "h" είναι η ευρετική λειτουργία μας, η εκτίμηση του κόστους που θα υπάρχει για να φτάσει ο πράκτορας στο στόχο. Το "φ" είναι το ποσό του συνόλου αυτών των δύο. Θα αποθηκεύσουμε κάθε έναν από αυτούς τους αριθμούς στους κόμβους μας. Χρησιμοποιώντας το φ, το γ και το χ εκτιμούμε τον αλγόριθμο A *. Έτσι μπορεί ο αλγόριθμος που κινεί τον πράκτορα να έχει μια πολύ καλή εικόνα του κόστους της κίνησης του πράκτορα από τον ένα κόμβο σε κάποιον άλλο και να μπορεί τελικά να υπολογίσει την βέλτιστη διαδρομή.

2.13 Κορυφαίες στιγμές της T.N.

2.13.1 Διαστημικά προγράμματα

Το πιο γνωστό πρόγραμμα της T.N. που χρησιμοποιεί η NASA είναι το «NASA's Remote Agent Program». Το συγκεκριμένο πρόγραμμα είναι μια εφαρμογή στην ουσία αυτόνομου σχεδιασμού και προγραμματισμού των λειτουργιών ενός διαστημοπλοίου. Οι ικανότητες αυτού του προγράμματος είναι ότι μπορεί να εντοπίσει, να κάνει διάγνωση των προβλημάτων και τέλος να επανακάμψει από αυτά.

2.13.2 Επίλυση εύκολων προβλημάτων

Ο πρώτος υπολογιστής ο οποίος νίκησε τον Garry Kasparov, παγκόσμιο πρωταθλητή στο σκάκι, με σκορ 3,5 – 2,5 ήταν ο Deep Blue της IBM σε φιλικό αγώνα. Ο Kasparov δήλωσε ότι αισθάνθηκε μία «νέα μορφή νοημοσύνης» πάνω από τη σκακιέρα. Το περιοδικό Newsweek περιέγραψε τον αγώνα σαν το «τελευταίο προπύργιο του εγκεφάλου». Αξίζει να σημειωθεί ότι η αξία των μετοχών της IBM αυξήθηκε κατά 18 δισεκατομμύρια δολάρια.

2.13.3 Αυτόνομος έλεγχος

1) «Οι υπολογιστές σήμερα είναι σε θέση να οδηγήσουν αυτοκίνητα χωρίς να υπάρχουν κατευθυντήριες γραμμές στο δρόμο. Αυτό συμβαίνει χρησιμοποιώντας ένα συνδυασμένο και προηγμένο σύστημα από βιντεοκάμερες, ραντάρ, λέιζερ και GPS τα οποία συγκεντρώνουν δεδομένα από το περιβάλλον. Ταυτόχρονα ισχυροί υπολογιστές εγκατεστημένοι στο ίδιο το όχημα επεξεργάζονται τα δεδομένα και οδηγούν το αυτοκίνητο. Σε έναν αγώνα δρόμου που πραγματοποιήθηκε στην έρημο της Νεβάδα και οργανώθηκε από την DARPA, 23 αυτοκίνητα – ρομπότ διένυσαν 211 χλμ. μέσα στην έρημο μέσα σε λιγότερο από 10 ώρες και αυτό είναι μόλις το πρώτο βήμα.» Science Illustrated Τεύχος 11, Φεβρουάριος 2006, σελ. 26-31

2.13.4 Διαγνωστικά συστήματα

Εφαρμογές ιατρικών διαγνώσεων βασισμένες σε ανάλυση πιθανοτήτων είναι ικανές να λειτουργήσουν στο επίπεδο ενός ειδικευμένου φυσικοθεραπευτή και σε αρκετές άλλες ειδικότητες. Το πρόγραμμα βασίζεται πάνω σε αναζήτηση κόμβων όπου ο ιατρός θα μπορεί να θέτει ερώτημα στον υπολογιστή και ο τελευταίος να του επιστρέφει μία επεξήγηση της διάγνωσης.

2.13.5 Ρομποτική

Πολλοί χειρούργοι σήμερα χρησιμοποιούν βοηθούς – ρομπότ στη μικροχειρουργική. Το HipNav είναι ένα σύστημα που χρησιμοποιεί τεχνικές ανάλυσης εικόνας μέσω υπολογιστή για να δημιουργήσει ένα τρισδιάστατο μοντέλο της εσωτερικής ανατομίας του ασθενή και στη συνέχεια χρησιμοποιεί ρομποτικό έλεγχο για να καθοδηγήσει την προσθήκη μιας οστεϊκής ενίσχυσης (π.χ. στο πόδι του ασθενούς).

2.14 Συμπερασματικά

Στο κεφάλαιο αυτό αναλύθηκε η επίλυση των προβλημάτων με αλγόριθμους αναζήτησης. Δείξαμε τον τρόπο με τον οποίο γίνεται ο καθορισμός του προβλήματος

και πώς πρέπει να λειτουργήσουμε για να το λύσουμε. Για να βρεθεί βέβαια η λύση αναλύεται η δομή του προβλήματος για να κατανοηθεί ο στόχος και να καταφέρουμε με αυτό τον τρόπο να αναπτύξουμε τον καλύτερο δυνατό αλγόριθμο αναζήτησης ο οποίος θα μπορέσει να λύσει τον γρίφο. Αφού κάναμε το σχεδιασμό αναλύουμε τα προβλήματα παιγνίων που λύνουν οι αλγόριθμοι αναζήτησης και δείχνουμε τη δομή τους. Στη συνέχεια αναφέραμε κάποια προβλήματα που έχει αναλάβει η τεχνητή νοημοσύνη να επιλύσει σε πραγματικό περιβάλλον και όχι σε περιβάλλον εργαστηρίου. Τα προβλήματα αυτά περιλαμβάνουν κυρίως προβλήματα δικτύων υπολογιστών και μεταφοράς δεδομένων, όπως για παράδειγμα μεταφορά αρχείων πολυμέσων.

Έπειτα γίνεται μια εκτενής αναφορά και ανάλυση των αλγόριθμων αναζήτησης και συγκεκριμένα των αλγόριθμων αναζήτησης κατά πλάτος, κατά βάθος, σε ορισμένο βάθος, βέλτιστης επέκτασης και αναζήτηση με τον αλγόριθμο A*.

Στο επόμενο κεφάλαιο θα αναλύσουμε και θα περιγράψουμε μια εφαρμογή επάνω στους αλγόριθμους αναζήτησης. Πιο συγκεκριμένα η εφαρμογή αυτή προσομοιώνει το περιβάλλον ενός λαβυρίνθου δύο διαστάσεων ο οποίος δημιουργείται τυχαία και περιλαμβάνει έναν κινητό πράκτορα ο οποίος προσπαθεί να φτάσει στον τερματισμό κάνοντας χρήση αλγορίθμου αναζήτησης με μνήμη των περασμάτων. Επίσης υπάρχει και ένα κινητό εμπόδιο που κάνει χρήση αλγορίθμου τυφλής αναζήτησης με σκοπό να εντοπίσει τον κινητό πράκτορα.

ΚΕΦΑΛΑΙΟ 3

Η ΕΦΑΡΜΟΓΗ ΚΑΙ Ο ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΤΗΣ

3.1 Εισαγωγή

Στο προηγούμενο κεφάλαιο αναλύσαμε θεωρητικά τους αλγόριθμους αναζήτησης. Μελετήσαμε τα προβλήματα παιγνίων, αναλύσαμε τη δομή τους και είδαμε πως μπορούν οι αλγόριθμοι αναζήτησης να βρουν εφαρμογή σε καθημερινές καταστάσεις και πιο συγκεκριμένα στα δίκτυα υπολογιστών και την μεταφορά αρχείων.

Στο τρίτο κεφάλαιο θα δείξουμε πρακτικά την λειτουργία των αλγόριθμων αναζήτησης. Δημιουργήσαμε μία εφαρμογή η οποία λειτουργεί βασισμένη στους αλγόριθμους αναζήτησης και προσομοιώνει έναν λαβύρινθο. Ο λαβύρινθος αυτός είναι δύο διαστάσεων και περιλαμβάνει έναν κινητό πράκτορα ο οποίος προσπαθεί να βρει την έξοδο του λαβυρίνθου και ένα κινητό εμπόδιο που στόχος του είναι να ανακόψει την πορεία του πράκτορα. Ο κινητός πράκτορας, αλλά και το εμπόδιο κάνουν χρήση αλγορίθμων αναζήτησης.

Στο κεφάλαιο αυτό δείχνουμε τον προγραμματισμό του λαβυρίνθου, το πώς σχεδιάστηκε, ποιες συναρτήσεις χρησιμοποιεί, περιγράφουμε τους αλγορίθμους που χρησιμοποιούν ο πράκτορας και το εμπόδιο, αναφερόμαστε στα προβλήματα που αντιμετωπίστηκαν κατά την δημιουργία της εφαρμογής και γενικότερα όλες τις λειτουργίες του λαβυρίνθου.

3.2 Ο αλγόριθμος κίνησης των κινητών πρακτόρων

Στο κεφάλαιο αυτό θα εξηγήσουμε τον τρόπο με τον οποίο κινούνται οι δύο κινητοί πράκτορες της εφαρμογής, ο ιππότης και ο δράκος. Θα πρέπει καταρχήν να αναφέρουμε ότι δεν κινούνται ταυτόχρονα στο περιβάλλον του λαβύρινθου, αλλά εναλλάξ. Πρώτα κινείται ο ιππότης και έπειτα ο δράκος.

Ο ιππότης σε γενικές γραμμές προσπαθεί να φτάσει όσο πιο γρήγορα στην πριγκίπισσα. Πρωταρχική του κίνηση δηλαδή είναι προς τα πάνω. Ξεκινά και ελέγχει αν έχει την δυνατότητα να κινηθεί προς τα πάνω. Αν μπορεί να κινηθεί, πηγαίνει όσο πιο ψηλά μπορεί, μέχρι να φτάσει σε αδιέξοδο. Τότε προσπαθεί να ξεφύγει από το αδιέξοδο πηγαίνοντας δεξιά. Θα συνεχίσει να κινείται δεξιά έως ότου βρει διέξοδο

προς τα επάνω. Αν στη δεξιά του πλευρά είναι αδιέξοδο ή βρίσκεται στην άκρη τότε ο ιππότης θα προσπαθήσει να πάει αριστερά. Θα συνεχίσει να πηγαίνει αριστερά μέχρι να βρει διέξοδο προς τα επάνω. Έτσι θα μπορέσει να συνεχίσει την πορεία προς τα επάνω και να φτάσει στην κορυφαία γραμμή του λαβυρίνθου όπου βρίσκεται η έξοδος. Αν δεν μπορέσει να παρακάμψει το εμπόδιο ή αν γυρίζει συνεχώς στην ίδια περιοχή τότε ο ιππότης κινείται προς τα κάτω.

Ο βρόγχος προς τα κάτω δεν περιλαμβάνει απλά την μετακίνηση του ιππότη αλλά και την παράκαμψη του αδιεξόδου που τυχόν έχει βρεθεί. Έτσι ο κινητός πράκτορας κινείται καθοδικά μέχρι να βρεθεί δίπλα του ένα τετράγωνο το οποίο να του επιτρέπει να κινηθεί προς τα δεξιά. Ο πράκτορας ελέγχει το τετράγωνο στο οποίο βρίσκεται και το διπλανό του και αν του επιτρέπεται η κίνηση κινείται δεξιά, μέχρι να βρεθεί κάποιος συνδυασμός τετραγώνων ο οποίος θα του επιτρέψει να κινηθεί ξανά προς τα κάτω και ξανά δεξιά. Έτσι μπορεί ο κινητός πράκτορας να αποφύγει τα τυχόν αδιέξοδα που θα βρεθούν μπροστά του.

Ο πράκτορας μπορεί και αναγνωρίζει το αν κινείται συνεχώς σε έναν ορισμένο χώρο. Το καταφέρνει αφού κρατά στη μνήμη του πόσες φορές ακριβώς πέρασε από κάποιο σημείο. Για την ακρίβεια υπολογίζεται ο αριθμός των ελέγχων που πραγματοποιεί για το αν μπορεί να συνεχίσει την κίνησή του ή όχι. Αυτό συμβαίνει γιατί μπορεί στην μέση της κίνησης προς τα πάνω για παράδειγμα να φτάσει σε κάποιο τετράγωνο που να του απαγορεύει την περαιτέρω κίνηση. Ο ιππότης θα συνεχίσει και θα κάνει ελέγχους για την κίνηση που μπορεί να πραγματοποιηθεί, έτσι μπορεί να φαίνεται ότι δεν κινείται αλλά να πραγματοποιούνται έλεγχοι, οπότε και αυξάνεται και ενημερώνεται ο Χάρτης Περασμάτων του λαβυρίνθου.

Αντίθετα ο δράκος δεν έχει σαν πρωταρχικό του σκοπό την κίνηση προς τα επάνω ή γενικότερα προς κάποια άλλη κατεύθυνση. Σκοπός του είναι να διατηρεί μια συνεχή κίνηση από τα αριστερά προς τα δεξιά του λαβυρίνθου έτσι ώστε να μπορέσει να ανακόψει την πορεία του ιππότη. Δεν κρατά κάποια μνήμη καθώς η αναζήτηση που εκτελείται είναι καθαρά αναζήτηση σε πλάτος. Ο δράκος χρησιμοποιεί μια παρόμοια τεχνική με τον ιππότη για να ξεφύγει από τις αδιέξοδους, χωρίς όμως να κρατά κάποια μνήμη σχετικά με τον αριθμό των φορών που προσπέλασε κάποιο σημείο.

Αφού προσπαθήσει δηλαδή να κινηθεί δεξιά και αριστερά χωρίς επιτυχία θα προσπαθήσει να κινηθεί προς τα επάνω και έπειτα ξανά δεξιά ή αριστερά. Πάντα

ελέγχοντας τα διπλανά τετράγωνα για το αν μπορεί να κινηθεί δεξιά ή αριστερά. Έπειτα εκτελείται σε τυχαίες χρονικές περιόδους και ένας βρόγχος κίνησης προς τα κάτω παρόμοιος με του ιππότη. Πηγαίνει δηλαδή προς τα κάτω μέχρι να βρει διέξοδο προς τα δεξιά και να ξεφύγει από τυχόν επιστροφή στην ίδια περιοχή αναζήτησης. Δεν κρατά σε καμία περίπτωση μνήμη για το από που έχει περάσει ή πόσες φορές, κάτι που γίνεται από την πλευρά του ιππότη.

Η μοναδική περίπτωση που θα αλλάξει ξαφνικά την πορεία του ο δράκος είναι να βρει τον ιππότη κοντά του. Οι δύο κινητοί πράκτορες έχουν την ιδιότητα να «βλέπουν» σε ακτίνα ενός τετραγώνου. Έτσι ο δράκος μπορεί για παράδειγμα να κινείται προς τα κάτω και δίπλα του να είναι σταματημένος ο ιππότης. Αυτόματα ο δράκος θα τον δει και θα σταματήσει την κίνησή του. Το μόνο που θα κάνει από εκεί και πέρα είναι να επιτεθεί στον ιππότη και να νικήσει. Ο ιππότης από την άλλη μεριά αν σε οποιαδήποτε στιγμή βρει τον δράκο, τότε σταματά την κίνησή του και κινείται σε μια οποιαδήποτε άλλη κατεύθυνση για να ξεφύγει. Πάντα υπάρχει η πιθανότητα βέβαια η κατεύθυνση στην οποία να προσπαθήσει να ξεφύγει ο ιππότης να είναι αδιέξοδος. Έτσι ο δράκος με το που θα έρθει η σειρά του (αφού όπως προαναφέραμε οι κινήσεις γίνονται εναλλάξ) θα επιτεθεί στον παίκτη και θα κερδίσει.

Συγκεντρωτικά μπορούμε να πούμε ότι ο αλγόριθμος κίνησης εκτελεί τα παρακάτω βήματα για την κίνησή του.

Βήμα 1ο:

Ξεκινώντας ο πράκτορας την εξερεύνηση του περιβάλλοντος του λαβύρινθου, πρέπει να κοιτάξει γύρω του και να μάθει τα τέσσερα (ή τρία αν βρίσκεται στην άκρη) τετράγωνα τα οποία είναι υποψήφια για να κινηθεί. Βλέπει το πάνω τετράγωνο και μαθαίνει το εικονίδιο που υπάρχει εκεί. Αν δεν έχει ξαναπάει εκεί και το εικονίδιο στο οποίο βρίσκεται σε συνδυασμό με το πιο πάνω εικονίδιο επιτρέπουν την κίνηση προς τα πάνω τότε μετακινείται ένα τετράγωνο προς τα πάνω, αλλιώς πηγαίνε στο βήμα 2. Η κίνηση προς τα πάνω συνεχίζεται όσο το επιτρέπει η κατασκευή του λαβυρίνθου.

Βήμα 2ο:

Αν ο πράκτορας κατά την κίνησή του προς τα επάνω βρει εμπόδιο τότε αλλάζει κατεύθυνση και προσπαθεί να πάει δεξιά. Ελέγχει το διπλανό τετράγωνο ποιο είναι και αν το εικονίδιο επιτρέπει την κίνηση προς τα δεξιά και αν δεν έχει περάσει από το συγκεκριμένο τετράγωνο πάνω από 30 φορές τότε πηγαίνει δεξιά. Αν δεν μπορεί να κινηθεί δεξιά ή έχει περάσει πάνω από 30 φορές τότε πηγαίνει στο βήμα 3. Αν πραγματοποιηθεί η κίνηση προς τα δεξιά τότε πηγαίνει στο βήμα 1.

Βήμα 3ο:

Αν για κάποιο λόγο ο πράκτορας δεν μπορέσει να πάει δεξιά τότε η επόμενη κίνησή του είναι αριστερά. Ελέγχει το διπλανό τετράγωνο ποιο είναι και αν το εικονίδιο επιτρέπει την κίνηση προς τα αριστερά και αν δεν έχει περάσει από το συγκεκριμένο τετράγωνο πάνω από 30 φορές τότε πηγαίνει αριστερά. Αν δεν μπορεί να κινηθεί ούτε αριστερά ή έχει περάσει από εκεί πάνω από 30 φορές τότε πηγαίνει στο βήμα 4. Αν πραγματοποιηθεί η κίνηση προς τα αριστερά πηγαίνει στο βήμα 1 αλλιώς πηγαίνει στο βήμα 4.

Βήμα 4ο:

Αν ο πράκτορας φτάσει στο βήμα 4 σημαίνει ότι βρίσκεται κοντά σε αδιέξοδο. Για να μπορέσει να ξεφύγει τότε στο 4ο βήμα κάνει μια «ιδιότροπη κίνηση» προς τα κάτω. Η ιδιοτροπία της κίνησης οφείλεται στο ότι πηγαίνει προς τα κάτω και αμέσως μετά δεξιά αν πάντα μπορεί να κινηθεί κατ'αυτό τον τρόπο. Θα προσπαθήσει να κινηθεί 5 φορές έτσι. Αν δεν μπορέσει να κινηθεί ή βρεθεί κάποιο τετράγωνο που έχει προσπελαστεί πάνω από 30 φορές τότε πηγαίνει στο βήμα 5 που είναι και το τελευταίο.

Βήμα 5ο:

Αν δεν μπορέσει ο πράκτορας να κινηθεί κάτω-δεξιά τότε η τελευταία λύση είναι να κινηθεί παρόμοια με το βήμα 4 δηλαδή κάτω και αριστερά. Στη συνέχεια είτε μπορέσει να κινηθεί είτε όχι, πηγαίνει στο βήμα 1.

3.3 Το interface του λαβύρινθου

Η εφαρμογή είναι βασισμένη στην δημιουργία φορμών. Με την εκκίνηση και πιο συγκεκριμένα τρέχοντας το αρχείο EXE του προγράμματος, αυτόματα φορτώνει η κύρια φόρμα της εφαρμογής στην οποία είναι γραμμένο το μεγαλύτερο μέρος του κώδικα. Ξεκινώντας εμφανίζεται ο λαβύρινθος, ο οποίος έχει ήδη δημιουργηθεί τυχαία, οι δύο κινητοί πράκτορες του προγράμματος, ο παίκτης (ιππότης) και ο δράκος καθώς επίσης και η έξοδος (πριγκίπισσα). Στο επάνω μέρος είναι ενεργοποιημένες οι επιλογές της εφαρμογής. Αυτές είναι: Παιχνίδι, Τεχνητή Νοημοσύνη, Χάρτης Περασμάτων, About. Η επιλογή Παιχνίδι έχει τρεις νέες επιλογές: Νέος Λαβύρινθος, Μηδενισμός, Exit. Η επιλογή Τεχνητή Νοημοσύνη έχει

δύο νέες επιλογές: Εκκίνηση και Χάρτης ΤΝ. Δεξιά υπάρχει το κουμπί εκκίνησης επίλυσης του λαβυρίνθου, αναγράφοντας «START THE HUNT». Το επιλεγμένο χρώμα του φόντου της εφαρμογής είναι το μαύρο καθώς κάνει πολύ καλή αντίθεση με το χρώμα της πίστας του λαβυρίνθου. Παρακάτω φαίνεται μια εικόνα του λαβύρινθου που δείχνει την εφαρμογή κατά την εκκίνησή της.

Το interface της εφαρμογής.

3.4 Η εκκίνηση

Φορτώνοντας την πρώτη φόρμα της εφαρμογής αυτόματα γίνονται οι παρακάτω διαδικασίες. Καταρχάς ενεργοποιούνται οι τέσσερις κινήσεις που μπορούν να κάνουν οι δύο κινητοί πράκτορες. Επάνω, κάτω, δεξιά και αριστερά. Φυσικά οι κινήσεις αυτές περιλαμβάνουν το σύνολο των ελέγχων που κάνουν οι πράκτορες για να κινηθούν στην επιθυμητή κατεύθυνση.

Στη συνέχεια μηδενίζονται όλες οι μεταβλητές, εκτός από αυτές τις οποίες χρησιμοποιεί το πρόγραμμα σαν τυχαίες. Αυτές είναι οι μεταβλητές μέσω των οποίων

επιλέγονται τα κατάλληλα εικονίδια για την κατασκευή του λαβύρινθου (καθώς ο λαβύρινθος θα πρέπει να είναι εντελώς τυχαίος) και οι μεταβλητές που αρχικοποιούν την θέση των δύο πρακτόρων καθώς και του τερματισμού. Τέλος καλούνται οι βιβλιοθήκες του Microsoft Media Player και αρχίζει να παίζει η μουσική υπόκρουση η οποία συνεχίζεται όσο είναι φορτωμένη η κύρια φόρμα.

Η εκκίνηση της επίλυσης του λαβύρινθου πραγματοποιείται κάνοντας κλικ ο χρήστης στο κουμπί που αναγράφει «START THE HUNT». Αυτόματα τρέχει ο κώδικας που ελέγχει τους δύο αλγόριθμους κίνησης των κινητών πρακτόρων. Το κύριο μέρος είναι ένας έλεγχος ροής "while" ο οποίος τρέχει συνέχεια έως ότου βρεθεί λύση στο λαβύρινθο ή ο ιππότης συναντηθεί με τον δράκο. Στο εσωτερικό του βρόγχου ροής οι δύο πράκτορες κινούνται με την σειρά. Πρώτα ξεκινά ο παίκτης και στη συνέχεια ο δράκος. Η σειρά κίνησης γίνεται εναλλάξ. Πάντα βέβαια υπάρχει η πιθανότητα κάποιος από τους δύο πράκτορες να χάσει την σειρά του. Αυτό γίνεται γιατί μπορεί να προσπαθήσει να κινηθεί σε κάποια κατεύθυνση όπου δεν θα επιτρέπεται, για παράδειγμα αν συναντήσει κάποιον τοίχο. Ο παρατηρητής της εφαρμογής θα παρατηρήσει μια στιγμιαία καθυστέρηση στην εναλλαγή της κίνησης, ένα «πάγωμα», όπου έπειτα θα ξανακινηθεί ο πράκτορας που κινήθηκε τελευταίος. Στο εσωτερικό των ελέγχων υπάρχουν και οι συναρτήσεις που ελέγχουν το πότε τελειώνει η εφαρμογή, πότε λύθηκε, πότε έχασε ο ιππότης, υπάρχουν οι συναρτήσεις όπου ο ιππότης μαθαίνει τον λαβύρινθο, οι στατιστικές συναρτήσεις και γενικότερα κάθε τύπου έλεγχος. Ακόμη και η συνάρτηση που καθυστερεί την κίνηση του παιχνιδιού για να μπορεί ο παρατηρητής να παρακολουθεί καλύτερα τη ροή των κινήσεων. Φυσικά μέσα στο βρόγχο απλά καλούνται οι συναρτήσεις. Η υλοποίησή τους γίνεται σε ξεχωριστό σημείο του κώδικα.

3.5 Το μενού «Νέος λαβύρινθος»

Το συγκεκριμένο μενού αποτελεί «παρακλάδι» του μενού «Παιχνίδι». Εδώ ο χρήστης μπορεί να επιλέξει ανά πάσα στιγμή να δημιουργηθεί ένας νέος λαβύρινθος. Αν δεν έχει ξεκινήσει την επίλυση του λαβυρίνθου, όλες οι μεταβλητές θα παραμείνουν μηδενισμένες, όλοι οι μετρητές μηδέν και η γνώση του λαβύρινθου μηδενική. Αν έχει γίνει η εκκίνηση της επίλυσης του λαβυρίνθου, τότε οι μεταβλητές

θα παραμείνουν ίδιες, οι μετρητές βημάτων ίδιοι. Θα εκκινήσουν με αρχικοποιημένες θέσεις μόνο οι πράκτορες και η έξοδος, καθώς βέβαια και ο νέος λαβύρινθος. Στην ουσία η επιλογή της συγκεκριμένης επιλογής αποτελεί μια επανεκκίνηση του λαβυρίνθου χωρίς να μηδενίζονται οι μεταβλητές.

3.6 Το μενού «Μηδενισμός»

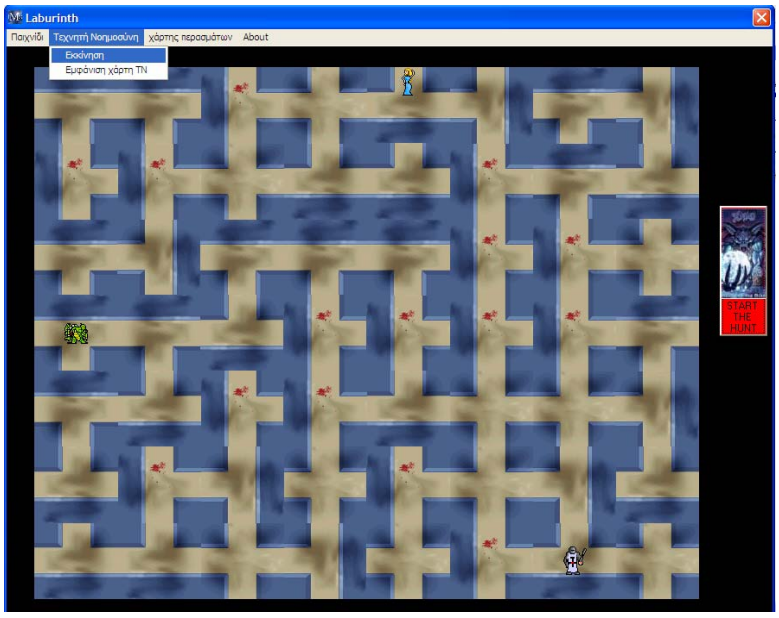
Εδώ ο χρήστης μπορεί όπως και με την πιο πάνω επιλογή να κάνει επανεκκίνηση του λαβυρίνθου. Φυσικά τα αποτελέσματα δεν είναι τα ίδια. Η επανεκκίνηση εδώ γίνεται μηδενίζοντας και αρχικοποιώντας όλες τις μεταβλητές που υπάρχουν στον κώδικα. Μηδενίζονται οι μετρητές καθώς επίσης και η γνώση που έχει ο πράκτορας για τον λαβύρινθο. Ταυτόχρονα γίνεται αρχικοποίηση των θέσεων των δύο πρακτόρων καθώς και αρχικοποίηση της θέσης της εξόδου. Στην ουσία επιλέγοντας το συγκεκριμένο μενού ο χρήστης ξεκινά ένα νέο παιχνίδι από την αρχή. Σαν να ξεκινά δηλαδή τώρα την εφαρμογή.

3.7 Το μενού «Exit»

Η λειτουργία της συγκεκριμένης επιλογής είναι απλή. Είναι η έξοδος της εφαρμογής. Επιλέγοντας ο χρήστης την συγκεκριμένη επιλογή όλες οι φόρμες που έχουν ανοιχτεί κλείνουν, κλείνοντας έτσι την εφαρμογή. Η εντολή αυτή γίνεται μέσω μιας απλής συνάρτησης η οποία όταν καλεστεί κλείνει ολόκληρη την εφαρμογή.

3.8 Το μενού «Τεχνητή Νοημοσύνη – Εκκίνηση»

Η συγκεκριμένη επιλογή δεν επιφέρει κάποια αλλαγή οπτική στην εφαρμογή. Αυτό το οποίο κάνει είναι να ξεκινήσει μια συνάρτηση η οποία τρέχει για όση ώρα τρέχει η εφαρμογή και σταματάει μόνο αν επιλεχθεί ο μηδενισμός ή να δημιουργηθεί κάποιος νέος λαβύρινθος. Σαν παράμετρος αυτής της συνάρτησης δίνεται η μεταβλητή που είναι αποθηκευμένη η θέση του ιππότη (pcpos). Η συνάρτηση αυτή είναι η LearnAImap (pcpos) και σκοπός της είναι να μαθαίνει τι βρίσκεται γύρω από τον ιππότη. Καταγράφει ποιο είναι το τετράγωνο στο οποίο βρίσκεται τη δεδομένη στιγμή ο ιππότης, το τετράγωνο μπροστά του, δεξιά και αριστερά του, καθώς και πίσω του. Στη συνέχεια καλείται η συνάρτηση AImap η οποία είναι στην ουσία ο χάρτης της τεχνητής νοημοσύνης. Θα πρέπει να σημειωθεί ότι η συνάρτηση LearnAImap (pcpos) καταγράφει τον αριθμό που αντιπροσωπεύεται σε κάθε εικονίδιο.



3.9 Το μενού «Τεχνητή Νοημοσύνη − Χάρτης ΤΝ»

Η συγκεκριμένη επιλογή είναι στην ουσία συνέχεια της προηγούμενης. Αφού έχει ενεργοποιηθεί η προηγούμενη επιλογή, υπάρχει όπως προαναφέρθηκε μια συνεχής καταγραφή. Τα δεδομένα της καταγραφής αυτής αποθηκεύονται σε ένα string το οποίο έχει το μέγεθος του λαβυρίνθου, δηλαδή 56 θέσεις. Στην κάθε θέση αποθηκεύεται το είδος του εικονιδίου που βλέπει ο ιππότης. Την αποθήκευση των δεδομένων αυτών αναλαμβάνει η συνάρτηση AImap η οποία διαβάζει τα δεδομένα από την LearnAImap (pcpos).

Η AImap είναι στην ουσία ένας χάρτης ο οποίος είναι ίδιος με τον λαβύρινθο. Φυσικά ο χάρτης αυτός είναι μηδενισμένος στην εκκίνηση της εφαρμογής και μηδενίζεται κάθε φορά που κάνει την επιλογή μηδενισμού ο χρήστης. Έτσι όταν κινείται ο ιππότης ενημερώνεται δυναμικά ο χάρτης και στη συνέχεια τα στοιχεία του string. Όταν ο χρήστης επιλέξει να εμφανιστεί ο χάρτης της τεχνητής νοημοσύνης, εμφανίζεται το string σε ένα ξεχωριστό παράθυρο. Τα στοιχεία που εμφανίζονται στο παράθυρο είναι με τη μορφή 7x8 και στην ουσία αντιπροσωπεύουν τον λαβύρινθο καθώς και ο λαβύρινθος είναι της μορφής 7x8. Ο χάρτης της τεχνητής νοημοσύνης εμφανίζεται με την μορφή ξεχωριστού παραθύρου το οποίο ελέγχεται από το κουμπί «ok». Όταν ο χρήστης επιλέξει το κουμπί «ok» η εφαρμογή συνεχίζει κανονικά τη ροή της. Το παράθυρο ενεργοποιείται από την συνάρτηση MsgBox s, vbOKOnly, "Χάρτης ΤΝ".

Παρακάτω φαίνεται ο Χάρτης της ΤΝ καθώς τρέχει το πρόγραμμα το λαβυρίνθου.

Εμφάνιση του Χάρτη της ΤΝ.

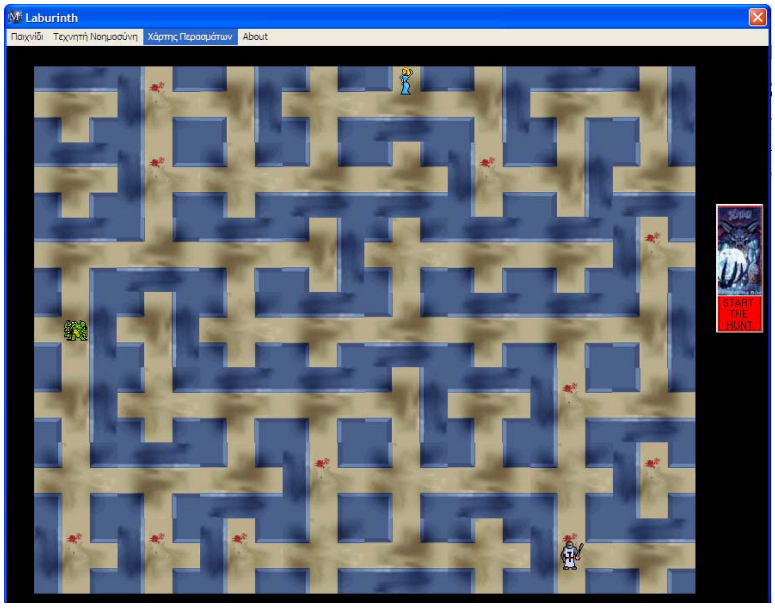
3.10 Το μενού «Χάρτης Περασμάτων»

Η επιλογή «Χάρτης Περασμάτων» είναι ένας μετρητής ο οποίος ενημερώνεται δυναμικά. Μετρά το πόσες φορές έκανε έλεγχο ο κινητός πράκτορας για να μπορέσει να κινηθεί στο επόμενο τετράγωνο. Φυσικά ο συγκεκριμένος χάρτης δείχνει μόνο τους ελέγχους που κάνει ο ιππότης.

Ο χάρτης αυτός είναι μία φόρμα σε αντίθεση με τον χάρτη της τεχνητής νοημοσύνης που είναι ένα string. Βέβαια η φόρμα αυτή δεν είναι τόσο απλή όσο η συνάρτηση MsgBox (s, vbOKOnly). Εδώ αντί να χρησιμοποιηθεί η τεχνική της δημιουργίας ενός string, χρησιμοποιείται τεχνική αντικειμένων. Αντί δηλαδή να είναι κάθε τετράγωνο του λαβυρίνθου ένα στοιχείο του string, τώρα στη φόρμα κάθε τετράγωνο του λαβυρίνθου αντιπροσωπεύεται από ένα αντικείμενο τύπου TextBox. Το πλεονέκτημα εδώ προγραμματιστικά είναι ότι μπορούν πολύ εύκολα να συνδεθούν τα αντικείμενα αυτά με τις υπόλοιπες φόρμες της εφαρμογής. Επίσης η

δυναμική ενημέρωση των “TextBoxes” είναι πολύ πιο εύκολη και αποτελεσματική. Ακόμη, τα “TextBoxes” σε μία φόρμα είναι ευανάγνωστα στον παρατηρητή της εφαρμογής, καθώς επίσης και οι συνεχείς αλλαγές των τιμών τους. Οι τιμές αυτές είναι μηδενισμένες κάθε φορά που ξεκινά ο λαβύρινθος, καθώς και όταν επιλέξει ο χρήστης την επιλογή «Μηδενισμός». Αν ο χρήστης επιλέξει την επιλογή «Νέος Λαβύρινθος» τότε οι τιμές παραμένουν αμετάβλητες. Το «πέρασμα» των τιμών στα αντικείμενα γίνεται μέσω της συνάρτησης showcounter( ). Η συνάρτηση αυτή διαβάζει το πόσοι έλεγχοι γίνονται από τον ιππότη κάθε φορά που έρχεται η κίνησή του και είτε κινείται είτε όχι προσθέτει τον κατάλληλο αριθμό στις τιμές των αντικειμένων “TextBoxes”.

Για να εμφανιστεί η φόρμα που δείχνει το χάρτη περασμάτων καλείται απλά η συνάρτηση Private Sub mnushowcountingmap_Click( ) η οποία ενεργοποιείται όταν ο χρήστης κάνει κλικ στο μενού «Χάρτης Περασμάτων». Έπειτα τρέχει ο κώδικας της συνάρτησης ο οποίος ενεργοποιεί την ιδιότητα Visible του αντικειμένου countermap και της θέτει την τιμή true, οπότε και εμφανίζεται η φόρμα που δείχνει το χάρτη περασμάτων και συνεχίζει να ενημερώνεται δυναμικά.



3.11 Ο σχεδιασμός και ο κώδικας της εφαρμογής

Ο λαβύρινθος δημιουργείται όταν ο χρήστης τρέξει το αρχείο εκκίνησης της εφαρμογής, το οποίο είναι το maze1.exe. Όπως προαναφέρθηκε, για να μπορέσει η εφαρμογή να τρέξει σωστά, γίνονται κάποιες συγκεκριμένες διαδικασίες κατά την εκκίνηση. Καταρχήν γίνεται ένας τυχαίος σχεδιασμός του λαβυρίνθου προς επίλυση. Ο σχεδιασμός αυτός γίνεται ως εξής: Ο λαβύρινθος αποτελείται από 56 συνολικά frames τα οποία έχουν το ίδιο μέγεθος μεταξύ τους και είναι έτσι τοποθετημένα ώστε να σχηματίζουν έναν πίνακα μεγέθους 8 γραμμών επί 7 στηλών. Στο κάθε frame αντιστοιχούν 5 διαφορετικά εικονίδια.

Τα εικονίδια για υλοποίηση του λαβύρινθου είναι τα παρακάτω:

1) Εικονίδιο No 1

Το εικονίδιο αυτό επιτρέπει την κίνηση προς όλες τις κατευθύνσεις αλλά και από όλες τις κατευθύνσεις αντίστοιχα. Δεν έχει κανένα αδιέξοδο.

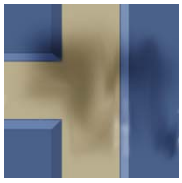
2) Εικονίδιο No 2

Το εικονίδιο αυτό επιτρέπει την κίνηση προς όλες τις κατευθύνσεις εκτός από την κίνηση από το συγκεκριμένο εικονίδιο προς τα αριστερά. Ταυτόχρονα απαγορεύει την κίνηση από το αριστερά τετράγωνο προς το συγκεκριμένο καθώς η αριστερή του πλευρά καταλήγει σε αδιέξοδο.

3) Εικονίδιο No 3

Το εικονίδιο αυτό επιτρέπει την κίνηση προς όλες τις κατευθύνσεις εκτός από την κίνηση από το συγκεκριμένο εικονίδιο προς τα επάνω. Ταυτόχρονα

απαγορεύει την κίνηση από το αμέσως πιο πάνω τετράγωνο προς το συγκεκριμένο καθώς η από πάνω του πλευρά καταλήγει σε αδιέξοδο.

4)  Εικονίδιο No 4

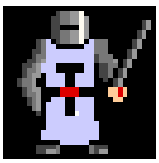
Το εικονίδιο αυτό επιτρέπει την κίνηση προς όλες τις κατευθύνσεις εκτός από την κίνηση από το συγκεκριμένο εικονίδιο προς τα δεξιά. Ταυτόχρονα απαγορεύει την κίνηση από το αμέσως πιο δεξιά τετράγωνο προς το συγκεκριμένο καθώς η δεξιά του πλευρά καταλήγει σε αδιέξοδο.

5) Εικονίδιο No 5

Το εικονίδιο αυτό επιτρέπει την κίνηση προς όλες τις κατευθύνσεις εκτός από την κίνηση από το συγκεκριμένο εικονίδιο προς τα κάτω. Ταυτόχρονα απαγορεύει την κίνηση από το αμέσως πιο κάτω τετράγωνο προς το συγκεκριμένο καθώς η κάτω του πλευρά καταλήγει σε αδιέξοδο.

Σε αυτό το σημείο πρέπει να σημειωθεί ότι η αρίθμηση που χρησιμοποιείται πιο πάνω, είναι η αντίστοιχη που χρησιμοποιεί η τεχνητή νοημοσύνη στον πίνακά της. Δηλαδή η αρίθμηση που φαίνεται στον πίνακα της τεχνητής νοημοσύνης όσον αφορά το τι έχει μάθει ο παίκτης , αντιστοιχεί με την πιο πάνω αρίθμηση των εικονιδίων. Επίσης η αρίθμηση που χρησιμοποιείται στον προγραμματισμό των κινήσεων του παίκτη και του Wumpus αντιστοιχεί στην πιο πάνω.

Τα εικονίδια του Παίκτη , Wumpus , εξόδου , κουμπιού εκκίνησης είναι τα εξής:

 Ο κινητός πράκτορας (Ιππότης).

Το κινητό εμπόδιο (Δράκος).

Ο στόχος – έξοδος του λαβύρινθου. Τι άλλο? Μια βασίλισσα.

Τα εικονίδια αυτά επιλέχθηκαν καθαρά για να μπορεί να είναι εύκολη η μεταξύ τους διάκριση αλλά και για να δημιουργείται μία ατμόσφαιρα περιπέτειας κατά την διάρκεια του παιχνιδιού.

Κουμπί έναρξης

Ειδικά για το εικονίδιο της έναρξης του παιχνιδιού χρησιμοποιήθηκε η συγκεκριμένη εικόνα εξαιτίας του συμβολισμού. Όπως το τέρας κρατάει στα χέρια του τον κόσμο και την μοίρα του, έτσι και η τεχνητή νοημοσύνη κρατάει στα χέρια της την μοίρα του Παίκτη και του Wumpus. Ταυτόχρονα η τεχνητή νοημοσύνη εν καιρώ θα είναι αυτή υπεύθυνη για ένα μεγάλο μέρος της μοίρας του δικού μας κόσμου.

Το ποιο εικονίδιο θα μπει σε κάθε ένα frame, εξαρτάται από τις τιμές των μεταβλητών που παίρνουν τυχαίες τιμές (ακριβώς για να διατηρηθεί ο τυχαίος χαρακτήρας του λαβυρίνθου). Οι μεταβλητές αυτές αρχικοποιούνται από έναν έλεγχο ροής for, ο οποίος αναλαμβάνει να βάλει τυχαία τα εικονίδια στα frames. Η διαδικασία έχει ως εξής:

```
countL = 0       «μηδενισμός της μεταβλητής countL»
   For countL = 0 To 55     «Για countL από 0 έως 55»
      Randomize         «δώσε τυχαίες τιμές»
      nisse = Int(Rnd * 5 + 1) «στρογγυλοποίησε σε ακέραια τιμή»
      Image1(countL) = LoadPicture(App.Path & "\res\" & nisse & ".bmp")
      Map(countL) = nisse   «δώσε την τιμή της μεταβλητής nisse στον χάρτη»
```

```
Next countL
```

Ο συγκεκριμένος κώδικας χρησιμοποιεί την μεταβλητή countL η οποία μετράει με βήμα 1 από το μηδέν ως το 55, δηλαδή για 56 φορές. Επίσης γίνεται χρήση της μεταβλητής nisse η οποία παίρνει κάθε φορά τυχαία τιμή από το 1 έως το 5. Στη συνέχεια περνιέται η τιμή της countL στο frame και έτσι το κάθε frame έχει και ένα εικονίδιο. Η διαδικασία αυτή ακολουθείται για όλα τα frames του λαβυρίνθου.

Όταν ξεκινά η εφαρμογή φορτώνεται και η μουσική του παιχνιδιού, η οποία παίζει μέσω του windows media player. Το media player είναι έτσι ρυθμισμένο ώστε να φορτώνει ένα συγκεκριμένο αρχείο μουσικής και να το επαναλαμβάνει όσο η εφαρμογή είναι ανοικτή. Ο τύπος αρχείου μουσικής που χρησιμοποιείται είναι αρχείο mp3 και συγκεκριμένα το princess_of_the_dawn.mp3.

Στη συνέχεια γίνεται η αρχικοποίηση των θέσεων των δύο κινητών πρακτόρων αλλά και της εξόδου. Η αρχικοποίηση των θέσεων έχει κάποια όρια τα οποία τηρούνται κάθε φορά που αρχικοποιείται ολόκληρος ο λαβύρινθος. Για την θέση του ιππότη υπάρχει ο περιορισμός ότι η θέση που θα ξεκινήσει θα είναι στο κάτω μέρος του λαβυρίνθου και πιο συγκεκριμένα στις θέσεις 48 – 55. Όσον αφορά το κινητό εμπόδιο του λαβυρίνθου, υπάρχει ο περιορισμός να ξεκινάει στο μέσο του λαβυρίνθου και πιο συγκεκριμένα σε μία από τις θέσεις 24 – 31. Το τέλος του λαβυρίνθου, δηλαδή η πριγκίπισσα, θα βρίσκεται στο επάνω μέρος και μάλιστα σε μία από τις θέσεις 0 – 7. Παρακάτω φαίνεται πώς ακριβώς γίνεται η διαδικασία αρχικοποίησης των θέσεων.

```
Dim i, j, g             «δήλωση μεταβλητών»
Randomize Timer   «Τυχαία τιμή στον μετρητή χρόνου»
i = Int(Rnd * 7)      «στρογγυλοποίηση της ακέραιας τιμής στο 7»
picend.Top = Image1(i).Top  «ευθυγράμμιση εικονιδίου»
picend.Left = Image1(i).Left + Image1(i).Width / 2 – picend.Width / 2
pcend = i
Randomize Timer
j = Int(Rnd * 7)
PC.Top = Image1(48 + j).Top + Image1(48 + j).Height / 2 – PC.Height / 2
```

```
PC.Left = Image1(48 + j).Left + Image1(48 + j).Width / 2 – PC.Width / 2
pcpos = 48 + j
Randomize Timer
g = Int(Rnd * 7)
Wumpus.Top = Image1(24 + g).Top + Image1(24 + g).Height / 2–Wumpus.Height / 2
Wumpus.Left = Image1(24 + g).Left + Image1(24 + g).Width / 2–Wumpus.Width / 2
pcwumpus = 24 + g
```

Η αρχικοποίηση των θέσεων των δύο κινητών πρακτόρων και της εξόδου περιλαμβάνει και την σωστή τοποθέτησή τους στο κέντρο του κάθε frame. Αυτό γίνεται υπολογίζοντας την απόσταση από την κορυφή και τα πλάγια του κάθε frame και με κατάλληλους μαθηματικούς υπολογισμούς τοποθετούνται στο κέντρο του. Η ευχρηστία της visual basic είναι τέτοια που μας επιτρέπει να αλλάξουμε το μέγεθος των εικονιδίων με ένα απλό drag και να τα κάνουμε τόσο μεγάλα ώστε να κινούνται ευδιάκριτα στο περιβάλλον της εφαρμογής.

Τέλος, ξεκινώντας η εφαρμογή, γίνεται αρχικοποίηση των πινάκων της τεχνητής νοημοσύνης καθώς και του χάρτη μέτρησης περασμάτων. Αυτό γίνεται ως εξής:

Για τον χάρτη της τεχνητής νοημοσύνης:

```
For i = 0 To 55
AImap(i) = 0    «μηδενισμός του χάρτη της Τεχ. Νοημ.»
Next i
```

Για τον χάρτη μέτρησης περασμάτων:

```
For o = 0 To 55
countmap(o) = 0      «μηδενισμός του χάρτη μέτρησης περασμάτων»
Next o
```

3.11.1 Οι έλεγχοι για την κίνηση των πρακτόρων

Για την κίνηση των κινητών πρακτόρων, πρέπει πρώτα να αναφερθούμε στους ελέγχους που γίνονται πριν γίνει κάποια κίνηση και έπειτα θα αναφερθούμε στη φιλοσοφία και υλοποίηση του αλγόριθμου κίνησης των πρακτόρων στο περιβάλλον της εφαρμογής. Θα ξεκινήσουμε την περιγραφή των ελέγχων του ιππότη.

Για να κινηθεί ο ιππότης προς τα πάνω εκτελείται ο παρακάτω κώδικας:

```
If pcpos > 7 Then       «αν το pcpos είναι μεγαλύτερο του 7»
   If Map(pcpos) <> 3 Then          «αν το εικονίδιο δεν είναι το 3»
      If (Map(pcpos – 8) <> 5) Then          «αν το πιο κάτω εικονίδιο δεν είναι το 5»
      If (pcpos – 8 <> pcwumpus) Then       «αν το εμπόδιο δεν είναι στο πιο κάτω frame»
pcpos = pcpos – 8               «να κινηθεί ο παίκτης ένα εικονίδιο κάτω»
steps = steps + 1       «να αυξηθεί ο μετρητής βημάτων κατά 1»
PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 – PC.Height / 2 «ευθυγράμμιση»
PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 – PC.Width / 2
End If
End If
End If
End If
```

Οι έλεγχοι που κάνει εδώ ο πράκτορας αφορούν τη θέση του στο λαβύρινθο, το είδος του εικονιδίου στο οποίο βρίσκεται καθώς επίσης και το είδος του εικονιδίου που βρίσκεται πάνω από αυτόν. Η μεταβλητή pcpos μας δείχνει τη θέση του ιππότη στο λαβύρινθο. Αν η θέση του ιππότη είναι στην κορυφαία σειρά, δηλαδή αν η μεταβλητή pcpos έχει τιμή από 0 μέχρι 7, τότε ο έλεγχος για κίνηση προς τα πάνω και γενικότερα η κίνηση προς τα πάνω σταματάει, γιατί πολύ απλά ο χαρακτήρας είναι στην άκρη του λαβυρίνθου και δεν μπορεί να προχωρήσει παραπέρα. Αν ο ιππότης δεν είναι στην κορυφαία σειρά, τότε ο έλεγχος συνεχίζεται για το αν μπορεί να κινηθεί. Ελέγχεται το εικονίδιο στο οποίο βρίσκεται ο ιππότης τη δεδομένη στιγμή καθώς επίσης και το αμέσως πιο πάνω εικονίδιο. Εάν το εικονίδιο που βρίσκεται ο ιππότης δεν έχει τιμή 3 καθώς επίσης και το επόμενο δεν έχει τιμή 5, τότε ο έλεγχος εγκρίνεται και ο χαρακτήρας μπορεί να κινηθεί. Αυτό σημαίνει ότι σε κανένα από τα δύο εικονίδια ο ιππότης δεν πέφτει πάνω σε τοίχο. Ο πράκτορας κινείται κανονικά και στη συνέχεια μπαίνει η διαδικασία ευθυγράμμισης του εικονιδίου του ιππότη στο κέντρο του frame.

Κίνηση του πράκτορα προς τα δεξιά πραγματοποιείται κατά ένα μεγάλο μέρος παρόμοια με την κίνηση του ιππότη προς τα πάνω. Ο κώδικας που εκτελείται είναι ο εξής:

```
If (pcpos <> 7 And pcpos <> 15 And pcpos <> 23 And pcpos <> 31 And pcpos <> 39
And pcpos <> 47 And pcpos <> 55) Then
   If (Map(pcpos) <> 4 And Map(pcpos + 1) <> 2) Then
         If pcpos > 7 Then
            If Map(pcpos – 8) = 5 Or (Map(pcpos) = 3) Then
            If (pcpos + 1 <> pcwumpus) Then
         pcpos = pcpos + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 – PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 – PC.Width / 2
End If
End If
End If
End If

If (pcpos <> 7 And pcpos <> 15 And pcpos <> 23 And pcpos <> 31 And pcpos <> 39
And pcpos <> 47 And pcpos <> 55) Then
   If (Map(pcpos) <> 4 And Map(pcpos + 1) <> 2) Then
      If (pcpos < 7) Then
 pcpos = pcpos + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 – PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 – PC.Width / 2
End If
End If
End If
End If
```

Οι έλεγχοι που κάνει εδώ ο πράκτορας αφορούν όπως είναι γνωστό τη θέση του στο λαβύρινθο, το είδος του εικονιδίου στο οποίο βρίσκεται καθώς επίσης και το είδος του εικονιδίου που βρίσκεται πάνω από αυτόν. Η μεταβλητή pcpos μας δείχνει τη θέση του ιππότη στο λαβύρινθο. Αν η θέση του ιππότη είναι στην άκρα δεξιά στήλη, δηλαδή αν η μεταβλητή pcpos έχει μία από τις τιμές, 7, 15, 23, 31, 39, 47, 55 τότε ο έλεγχος για κίνηση προς τα δεξιά σταματάει, γιατί πολύ απλά ο χαρακτήρας είναι στην άκρη του λαβυρίνθου και δεν μπορεί να προχωρήσει παραπέρα. Αν ο

ιππότης δεν είναι στην άκρα δεξιά στήλη, τότε ο έλεγχος συνεχίζεται για το αν μπορεί να κινηθεί. Ελέγχεται το εικονίδιο στο οποίο βρίσκεται ο ιππότης τη δεδομένη στιγμή καθώς επίσης και το αμέσως πιο πάνω εικονίδιο. Εάν το εικονίδιο που βρίσκεται ο ιππότης δεν έχει τιμή 4 καθώς επίσης και το επόμενο δεν έχει τιμή 2, τότε ο έλεγχος εγκρίνεται και ο χαρακτήρας μπορεί να κινηθεί. Αυτό σημαίνει ότι σε κανένα από τα δύο εικονίδια ο ιππότης δεν πέφτει πάνω σε τοίχο ή γενικότερα σε αδιέξοδο. Επίσης, ένας τελευταίος έλεγχος που γίνεται είναι αν στο επόμενο τετράγωνο βρίσκεται ο δράκος ή όχι. Αν δεν βρίσκεται, τότε ο πράκτορας κινείται κανονικά και στη συνέχεια μπαίνει η διαδικασία ευθυγράμμισης του εικονιδίου του ιππότη στο κέντρο του frame.

Ο παραπάνω κώδικας περιλαμβάνει δύο μικρούς ελέγχους οι οποίοι ελέγχουν αν ο πράκτορας βρίσκεται στην πρώτη γραμμή ή όχι. Αν βρίσκεται στην πρώτη γραμμή, τότε η κίνηση γίνεται προς τα δεξιά, χωρίς να υπάρχει ενδιαφέρον για τα από πάνω τετράγωνα. Αν όμως ο πράκτορας κινείται από την δεύτερη γραμμή και κάτω, δηλαδή σε frames με τιμή μεγαλύτερη του 7, τότε ελέγχει και τα πιο πάνω τετράγωνα. Αν κατά τη διάρκεια της κίνησης δεξιά βρεθεί κάποιο από τα πιο πάνω τετράγωνα να είναι τέτοιο, ώστε σε συνδυασμό με το σημείο που βρίσκεται ο πράκτορας να υπάρχει η δυνατότητα μετακίνησης προς τα πάνω, τότε ο πράκτορας θα διακόψει την κίνησή του προς τα δεξιά. Στη συνέχεια θα βγει από τον βρόγχο κίνησης προς τα δεξιά και θα προτιμήσει να κινηθεί προς τα πάνω.

Αντίστοιχη είναι και η κίνηση προς τα αριστερά. Ο αντίστοιχος κώδικας που εκτελείται είναι:

```
If (pcpos <> 0 And pcpos <> 8 And pcpos <> 16 And pcpos <> 24 And pcpos <> 32 And pcpos <> 40 And pcpos <> 48) Or (pcpos = 7 Or pcpos = 15 Or pcpos = 23 Or pcpos = 31 Or pcpos = 39 Or pcpos = 47 Or pcpos = 55) Then
        If Map(pcpos) <> 2 Then
        If Map(pcpos – 1) <> 4 Then
        If pcpos – 1 <> pcwumpus Then
  pcpos = pcpos – 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 – PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 – PC.Width / 2
End If
End If
End If
```

```
End If
```

Οι έλεγχοι που κάνει εδώ ο πράκτορας αφορούν και πάλι τη θέση του στο λαβύρινθο, το είδος του εικονιδίου στο οποίο βρίσκεται καθώς επίσης και το είδος του εικονιδίου που βρίσκεται πάνω από αυτόν. Η μεταβλητή pcpos μας δείχνει τη θέση του ιππότη στο λαβύρινθο. Αν η θέση του ιππότη είναι στην άκρα αριστερή στήλη δηλαδή αν η μεταβλητή pcpos παίρνει τιμές 0, 8, 16, 24, 32, 40, 48, 56 τότε ο έλεγχος για κίνηση προς τα αριστερά και γενικότερα η κίνηση προς τα αριστερά σταματάει, γιατί πολύ απλά ο χαρακτήρας είναι στην άκρη του λαβυρίνθου και δεν μπορεί να προχωρήσει παραπέρα. Αν ο ιππότης δεν είναι στην άκρα αριστερή στήλη, τότε ο έλεγχος συνεχίζεται για το αν μπορεί να κινηθεί. Ελέγχεται το εικονίδιο στο οποίο βρίσκεται ο ιππότης τη δεδομένη στιγμή καθώς επίσης και το αμέσως πιο δίπλα εικονίδιο προς τα αριστερά. Εάν το εικονίδιο που βρίσκεται ο ιππότης δεν έχει τιμή 2 καθώς επίσης και το επόμενο δεν έχει τιμή 4, τότε ο έλεγχος εγκρίνεται και ο χαρακτήρας μπορεί να κινηθεί. Αυτό σημαίνει ότι σε κανένα από τα δύο εικονίδια ο ιππότης δεν πέφτει πάνω σε τοίχο. Ο πράκτορας κινείται κανονικά και στη συνέχεια μπαίνει η διαδικασία ευθυγράμμισης του εικονιδίου του ιππότη στο κέντρο του frame. Επίσης υπάρχει και ένας μικρός έλεγχος για το αν στο τετράγωνο αριστερά υπάρχει ο δράκος ή όχι.

Όσον αφορά την κίνηση προς τα κάτω ισχύουν σχεδόν οι ίδιοι έλεγχοι που ισχύουν και στις υπόλοιπες κινήσεις. Ο κώδικας που τρέχει είναι:

```
If pcpos < 48 Then
    If Map(pcpos) <> 5 Then
        If Map(pcpos + 8) <> 3 Then
            pcpos = pcpos + 8
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 – PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 – PC.Width / 2
        End If
    End If
End If
```

Για ακόμη μια φορά οι έλεγχοι που κάνει εδώ ο πράκτορας αφορούν τη θέση του στο λαβύρινθο, το είδος του εικονιδίου στο οποίο βρίσκεται καθώς επίσης και το είδος του εικονιδίου που βρίσκεται κάτω από αυτόν. Η μεταβλητή pcpos μας δείχνει τη θέση του ιππότη στο λαβύρινθο. Αν η θέση του ιππότη είναι στην τελευταία σειρά, δηλαδή αν η μεταβλητή pcpos έχει τιμή από 48 μέχρι 55, τότε ο έλεγχος για κίνηση προς τα κάτω και γενικότερα η κίνηση προς τα κάτω σταματάει, γιατί πολύ απλά ο χαρακτήρας είναι στην άκρη του λαβυρίνθου και δεν μπορεί να προχωρήσει παραπέρα. Αν ο ιππότης δεν είναι στην τελευταία σειρά, τότε ο έλεγχος συνεχίζεται για το αν μπορεί να κινηθεί. Ελέγχεται το εικονίδιο στο οποίο βρίσκεται ο ιππότης τη δεδομένη στιγμή καθώς επίσης και το εικονίδιο που βρίσκεται ακριβώς από κάτω. Εάν το εικονίδιο που βρίσκεται ο ιππότης δεν έχει τιμή 5 καθώς επίσης και το επόμενο δεν έχει τιμή 3, τότε ο έλεγχος εγκρίνεται και ο χαρακτήρας μπορεί να κινηθεί. Αυτό σημαίνει ότι σε κανένα από τα δύο εικονίδια ο ιππότης δεν πέφτει πάνω σε τοίχο. Ο πράκτορας κινείται κανονικά και στη συνέχεια μπαίνει η διαδικασία ευθυγράμμισης του εικονιδίου του ιππότη στο κέντρο του frame.

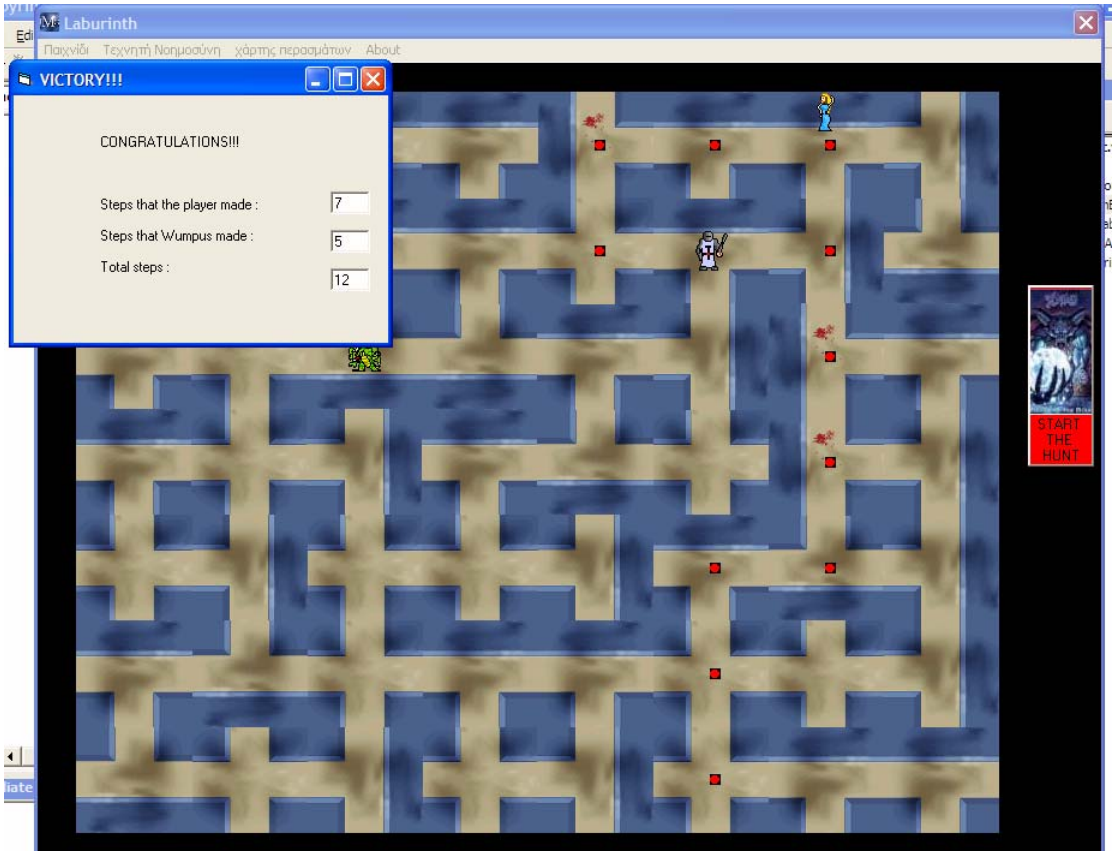
3.11.2 Τερματισμός του παιχνιδιού

Όσον αφορά τον τερματισμό του παιχνιδιού υπάρχουν δύο περιπτώσεις. Πρώτον να φτάσει ο παίκτης στον τερματισμό ή δεύτερον να σκοτωθεί ο ιππότης από τον δράκοντα. Για καθεμιά από τις δύο περιπτώσεις υπάρχει και η αντίστοιχη φόρμα.

Σε περίπτωση που ο ιππότης βρει τελικά την έξοδο, φορτώνεται η φόρμα FormEnd. Στην συγκεκριμένη φόρμα εμφανίζονται τρία πεδία. Τα πεδία αυτά είναι στην ουσία TextBoxes τα οποία ενημερώνονται κατά τη διάρκεια της εφαρμογής. Στο πρώτο πεδίο αναγράφονται τα συνολικά βήματα που έκανε ο ιππότης από τη στιγμή που έγινε η εκκίνηση του λαβυρίνθου, στο δεύτερο πεδίο τα συνολικά βήματα που έκανε ο δράκοντας και τέλος το τρίτο πεδίο μας εμφανίζει τα συνολικά βήματα που έγιναν μέχρι να φτάσει ο κινητός πράκτορας στην έξοδο.

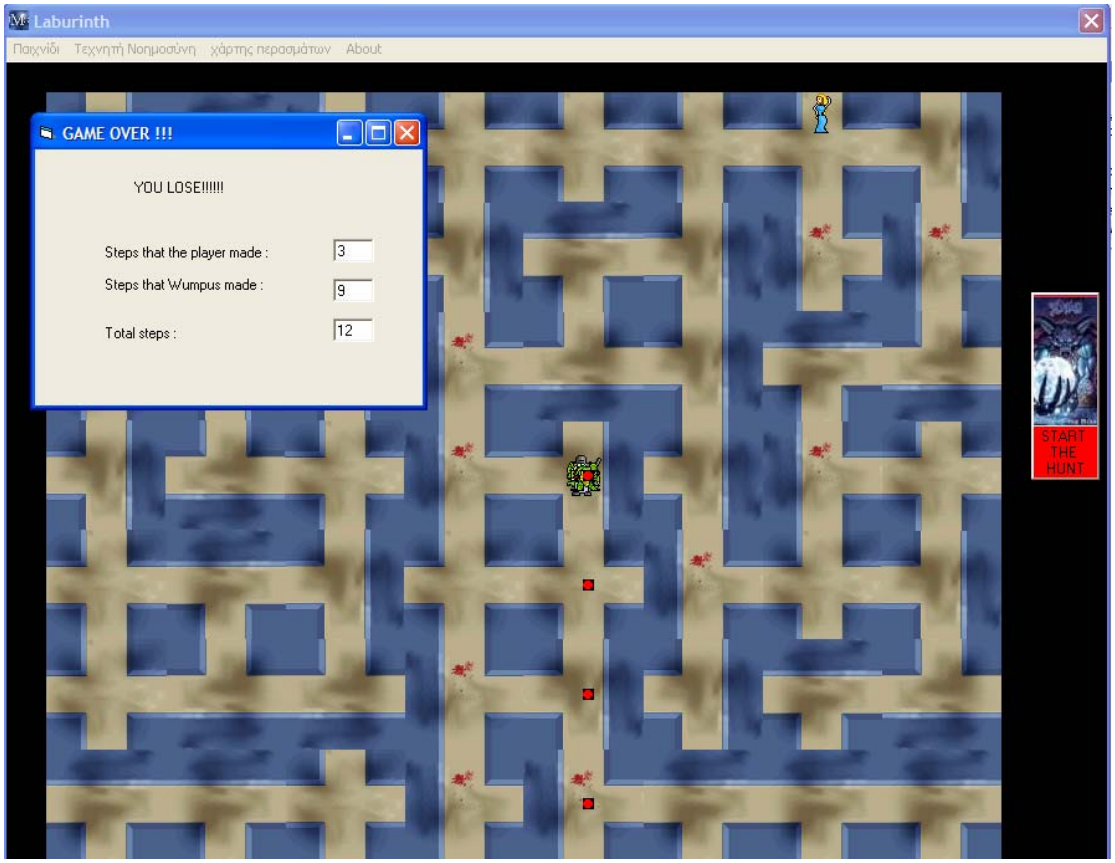
Οι τιμές της φόρμας μηδενίζονται όπως όλες οι μεταβλητές όταν επιλεχθεί από τον χρήστη η επιλογή «Μηδενισμός». Αντίθετα όταν ο χρήστης επιλέξει να γίνει απλά επανεκκίνηση του λαβυρίνθου, οι τιμές των μεταβλητών δεν μηδενίζονται.

Παρακάτω φαίνεται ένα δείγμα της φόρμας όταν ο ιππότης βρει την έξοδο.



Από την άλλη πλευρά όταν ο δράκοντας καταφέρει να εμποδίσει την κίνηση του ιππότη προς τον τερματισμό, φορτώνεται μία άλλη φόρμα και συγκεκριμένα η GAMEOVER. Εδώ τα μηνύματα που αναγράφονται στην φόρμα ενημερώνουν τον παρατηρητή ότι ο ιππότης έχασε. Χρησιμοποιώντας και πάλι τα TextBoxes εμφανίζονται τρεις τιμές στη φόρμα. Οι κινήσεις του ιππότη, του δράκοντα και τέλος οι συνολικές κινήσεις που έγιναν μέχρι να χάσει ο ιππότης. Και εδώ μηδενίζονται οι τιμές των μεταβλητών όταν ο χρήστης επιλέξει το μενού «Μηδενισμός» ενώ οι τιμές παραμένουν οι ίδιες όταν ο χρήστης επιλέξει την επανεκκίνηση της εφαρμογής μέσω του μενού «Νέος Λαβύρινθος».

Παρακάτω φαίνεται ένα δείγμα της συγκεκριμένης φόρμας όταν τρέχει η εφαρμογή.



3.12 Συναρτήσεις της εφαρμογής

Εδώ θα προσπαθήσουμε να αναλύσουμε τις συναρτήσεις που χρησιμοποιούνται από την εφαρμογή για την εκτέλεση κάποιων διαδικασιών. Οι διαδικασίες αυτές μπορεί να είναι έλεγχοι για το αν ο παίκτης έφτασε στην έξοδο, έλεγχοι για να βρει ο δράκος τον ιππότη, είναι συναρτήσεις που αναλαμβάνουν να αποθηκεύσουν τιμές σε μεταβλητές ή ακόμη και συναρτήσεις που καθυστερούν την ταχύτητα που εκτελούνται οι βρόγχοι της εφαρμογής. Θα πρέπει να αναφερθεί ότι οι συναρτήσεις καλούνται από το κύριο μέρος του κώδικα απλά με το όνομά τους, ενώ η υλοποίησή τους γίνεται σε ξεχωριστό μέρος του κώδικα, χάριν ευκολίας.

Ξεκινώντας θα αναλύσουμε δύο συναρτήσεις οι οποίες χρησιμοποιούνται από τον ιππότη για να ελέγχει αν όταν κινείται στην επάνω σειρά του λαβυρίνθου είναι

δίπλα του η έξοδος ή όχι. Αυτό συμβαίνει γιατί και ο ιππότης και ο δράκος έχουν την ικανότητα να μπορούν να δουν ένα τετράγωνο σε όλες τις κατευθύνσεις γύρω τους.

3.12.1 Οι συναρτήσεις lookright & lookleft

Οι δύο συναρτήσεις με τις οποίες ο ιππότης ελέγχει αν δεξιά ή αριστερά του βρίσκεται η έξοδος είναι η lookright( ) και η lookleft( ). Ο κώδικας που εκτελείται είναι ο παρακάτω:

```
Public Sub lookleft( )

If pcpos <> 0 And pcpos <> 8 And pcpos <> 16 And pcpos <> 24 And pcpos <> 32
And pcpos <> 40 And pcpos <> 48 Then
        If Map(pcpos) <> 2 Then
        If Map(pcpos – 1) <> 4 Then
        If pcpos – 1 <> pcwumpus Then
        If pcpos – 1 = pcend Then
pcpos = pcpos – 1
        PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 – PC.Height / 2
        PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 – PC.Width / 2
        End If
        End If
        End If
        End If
End If
counttimes
showcounter
CheckEnd
End Sub
```

Για την παραπάνω συνάρτηση γίνεται έλεγχος αριστερά του ιππότη για την ύπαρξη ή όχι της εξόδου. Καταρχήν, ο πρώτος έλεγχος με την if γίνεται για να ελεγχθεί σε ποια στήλη βρίσκεται ο κινητός πράκτορας. Για να πραγματοποιηθεί ο έλεγχος δεν θα πρέπει να βρίσκεται ο ιππότης στην άκρα αριστερή στήλη, γιατί πολύ

απλά δεν υπάρχει άλλο τετράγωνο πιο αριστερά για να ελέγξει. Στη συνέχεια γίνεται έλεγχος για το αν το τετράγωνο στο οποίο βρίσκεται ο ιππότης σε συνδυασμό με αυτό στα αριστερά του, δίνει τη δυνατότητα να πραγματοποιηθεί κίνηση. Αν η έξοδος βρίσκεται ακριβώς δίπλα μεν αλλά ο ιππότης δεν μπορεί να κινηθεί, τότε δεν υπάρχει νόημα να συνεχιστεί η διαδικασία. Ο ιππότης θα πρέπει να βρει άλλο δρόμο για να φτάσει στην πριγκίπισσα. Έπειτα πραγματοποιείται έλεγχος για το αν ο δράκος βρίσκεται στο διπλανό τετράγωνο. Ακόμη και αν η έξοδος είναι δίπλα από τον ιππότη, αν στο ίδιο τετράγωνο βρεθεί και ο δράκος, θα πρέπει ο ιππότης να μην πέσει επάνω του. Αφού πραγματοποιηθούν όλοι αυτοί οι έλεγχοι γίνεται και ο τελευταίος. Είναι η πριγκίπισσα στο διπλανό τετράγωνο? Αν είναι τότε άμεσα ο ιππότης κινείται αριστερά και έλυσε τον λαβύρινθο. Τέλος καλούνται οι συναρτήσεις CheckEnd, showcounter, counttimes για τις οποίες θα μιλήσουμε παρακάτω.

```
Public Sub lookright( )

If pcpos <> 7 And pcpos <> 15 And pcpos <> 23 And pcpos <> 31 And pcpos <> 39
And pcpos <> 47 And pcpos <> 55 Then
        If Map(pcpos) <> 4 Then
        If Map(pcpos + 1) <> 2 Then
        If pcpos + 1 <> pcwumpus Then
        If pcpos + 1 = pcend Then
pcpos = pcpos + 1
        PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 – PC.Height / 2
        PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 – PC.Width / 2
        End If
        End If
        End If
        End If
End If
trace
counttimes
showcounter
CheckEnd
End Sub
```

Η συνάρτηση lookright( ) είναι στην λειτουργία της ίδια με την lookleft( ) με τη μόνη διαφορά ότι οι έλεγχοι γίνονται στο τετράγωνο δεξιά του ιππότη. Ο κινητός πράκτορας ελέγχει για το αν βρίσκεται στην άκρα δεξιά στήλη του λαβυρίνθου, αν μπορεί να κινηθεί δεξιά, αν δεν βρίσκεται ο δράκος δεξιά του, αν είναι η πριγκίπισσα δίπλα του. Αν όλα αυτά ισχύουν τότε κινείται δεξιά και τελειώνει η αναζήτηση. Τέλος καλούνται και οι συναρτήσεις counttimes, showcounter, CheckEnd.

3.12.2 Η συνάρτηση counttimes( )

Είναι μία από τις απλούστερες συναρτήσεις της εφαρμογής. Αναλαμβάνει να αποθηκεύσει το πόσες φορές πέρασε από κάποιο τετράγωνο ο ιππότης. Ο κώδικάς της είναι:

```
Public Sub counttimes( )
countmap(pcpos) = countmap(pcpos) + 1
End Sub
```

Στην ουσία καλείται η συνάρτηση στην οποία αποθηκεύονται οι τιμές. Έτσι κάθε φορά που ο ιππότης περνά από κάποιο σημείο, καλείται η counttimes( ) και στη συνέχεια μέσω αυτής η countmap(pcpos), αυξάνει τον μετρητή. Όπου pcpos είναι η θέση του ιππότη.

3.12.3 Η συνάρτηση showcounter( )

Αναλαμβάνει να αποθηκεύσει στα TextBoxes της φόρμας countermap τις τιμές οι οποίες είναι αποθηκευμένες ήδη στη συνάρτηση countmap( ).

Εδώ είναι ένα απλό πέρασμα των τιμών στα TextBoxes της φόρμας. Η if ελέγχει αν ο παρατηρητής έχει επιλέξει να εμφανιστεί ο πίνακας μέτρησης των περασμάτων. Αν αυτό ισχύει, τότε άμεσα ενημερώνονται όλα τα TextBoxes, τα οποία παίρνουν τις τιμές που έχουν ήδη αποθηκευτεί στην countmap( ). Οι αριθμοί που εμφανίζονται είναι τα αριθμημένα frames της εφαρμογής. Η ενημέρωση είναι συνεχής παρόλο που δεν φαίνεται στον κώδικα. Αυτό συμβαίνει γιατί η συνάρτηση showcounter( ) καλείται κάθε φορά που κινούνται είτε ο ιππότης είτε ο δράκος.(βλ. παράρτημα για τον κώδικα)

3.12.4 Η συνάρτηση trace( )

Η συνάρτηση trace( ) δεν επηρεάζει την κίνηση των κινητών πρακτόρων, ούτε και αποθηκεύει μεταβλητές. Σκοπός της είναι να εμφανίσει, κάθε φορά που κινείται ο ιππότης, ένα ίχνος σε κάθε τετράγωνο που περνά. Εμφανίζει ένα μικρό κόκκινο εικονίδιο στο κέντρο του τετραγώνου που περνά ο κινητός πράκτορας.

3.13 Προβλήματα που αντιμετωπίστηκαν

Κατά την διάρκεια υλοποίησης του λαβύρινθου εμφανίστηκαν κάποια προβλήματα. Τα προβλήματα αυτά άλλοτε δεν άφηναν το πρόγραμμα να τρέξει καθόλου, άλλοτε δημιουργούσαν ενοχλητικές καταστάσεις. Τα προβλήματα αυτά εμφανίστηκαν στην πρώτη έκδοση του παιχνιδιού και στις επόμενες διορθώθηκαν ή ακόμη και βελτιώθηκε η απόδοσή τους. Τα κύρια προβλήματα που εμφανίστηκαν και αντιμετωπίστηκαν ήταν τα παρακάτω:

3.13.1 Μέγεθος του λαβύρινθου

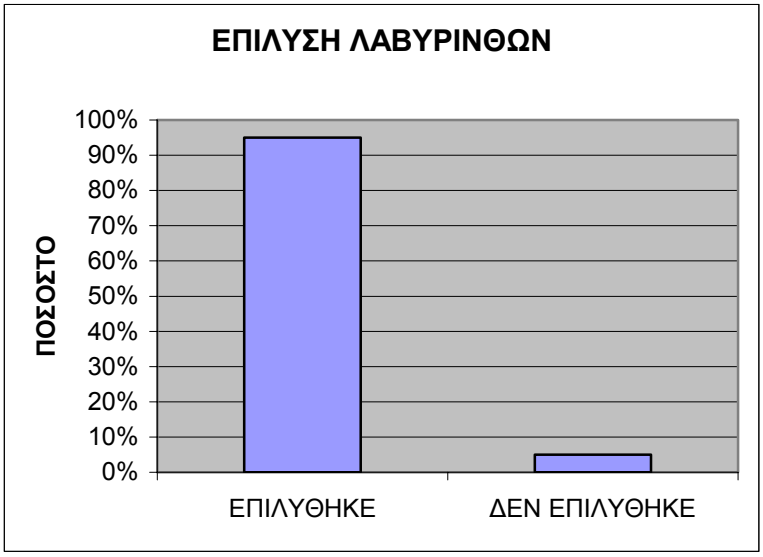
Στις πρώτες εκδόσεις του λαβύρινθου το μέγεθός του περιοριζόταν στο ένα τέταρτο περίπου της οθόνης. Στην τελευταία έκδοση το πρόβλημα αυτό αντιμετωπίστηκε και ο λαβύρινθος καταλαμβάνει το 80% περίπου της οθόνης. Για να αντιμετωπιστεί το πρόβλημα μεγεθύνθηκαν τα frames που αποτελούν τον λαβύρινθο. Μεγάλη προσοχή δόθηκε να μην υπερκαλύψει ο λαβύρινθος τα κουμπιά επιλογής.

3.13.2 Εγκλωβισμός του παίκτη

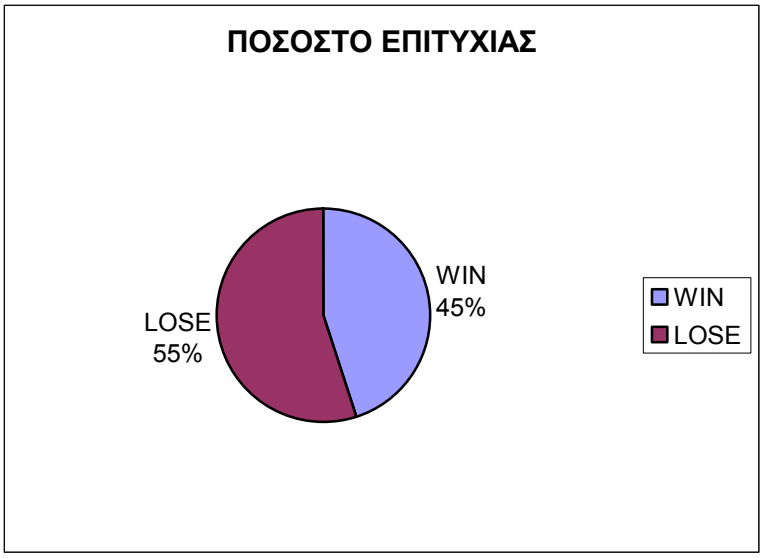
Πολλές φορές εξαιτίας του γεγονότος ότι ο λαβύρινθος δημιουργείται τυχαία ο παίκτης εγκλωβιζόταν σε μια συγκεκριμένη περιοχή του λαβύρινθου. Αυτό γινόταν εξαιτίας του προσανατολισμού της αναζήτησης. Ο παίκτης πήγαινε όσο το δυνατόν πιο ψηλά μπορούσε ώσπου να βρει αδιέξοδο. Όταν δεν μπορούσε να πάει πιο ψηλά προσπαθούσε να πάει δεξιά και αν πάλι δεν μπορούσε να προχωρήσει πήγαινε αριστερά. Στη συνέχεια αν δεν μπορούσε να πάει άλλο αριστερά αλλά ούτε και πάνω, τότε ξαναπήγαινε δεξιά οπότε και εγκλωβιζόταν.

Το πρόβλημα αυτό αντιμετωπίστηκε προσθέτοντας στον βρόγχο της κίνησης προς τα δεξιά να ψάχνει και να βρίσκει όχι μόνο διέξοδο προς τα πάνω αλλά και προς

τα κάτω. Επίσης στον βρόγχο της κίνησης προς τα κάτω προστέθηκε η δυνατότητα να ψάχνει για διέξοδο προς τα δεξιά. Έτσι επιτεύχθηκε η εκμηδένιση του εγκλωβισμού. Βεβαίως, δεν είναι σίγουρο κατά 100% ότι θα λυθεί ο λαβύρινθος. Πάντως, στατιστικά πάνω από το 95% των τυχαίων λαβύρινθων υπολογίστηκε ότι είτε βρέθηκε έστω και μετά από πολύ ώρα η λύση είτε ο παίκτης νικήθηκε από το Wumpus. Παρακάτω φαίνεται σε διάγραμμα το ποσοστό επιλυμένων ή όχι λαβυρίνθων. Στο γράφημα 2 φαίνεται το ποσοστό των λαβυρίνθων που υπήρξε τερματισμός είτε σαν νίκη του παίκτη, είτε σαν ήττα του.



Γράφημα 1



Γράφημα 2

3.13 3 Ταχύτητα βρόγχων

Ακόμη ένα πρόβλημα ήταν η ταχύτητα με την οποία κινούνταν ο παίκτης και το Wumpus μέσα στον λαβύρινθο. Κινούνταν με τόσο μεγάλη ταχύτητα που με δυσκολία μπορούσαν να διακριθούν μέσα στον λαβύρινθο.

Το πρόβλημα αυτό αντιμετωπίστηκε υλοποιώντας μία συνάρτηση pause( ) και μία συνάρτηση tmrpause_Timer( ). Η δεύτερη συνάρτηση κρατάει τον χρόνο σε δευτερόλεπτα, ενώ η πρώτη ελέγχει το αν ο χρόνος που μετράει η δεύτερη έχει ξεπεράσει την προκαθορισμένη χρονική στιγμή ή όχι. Αν την ξεπεράσει τότε αφήνει τον βρόγχο να ξανατρέξει. Έτσι καθυστερείται η εκτέλεση του βρόγχου και σαν αποτέλεσμα το Wumpus και ο παίκτης κινούνται όσο αργά θέλει ο προγραμματιστής να κινούνται.

3.13.4 Κίνηση στην φόρμα About

Για να κινηθούν η φωτογραφίες με τα γράμματα χρησιμοποιήθηκε αντίστοιχα η συνάρτηση timer1_Timer( ). Η τελευταία αναλαμβάνει να μετακινεί κατά έναν προκαθορισμένο αριθμό pixel τα γράμματα και τις εικόνες, ανά ένα δευτερόλεπτο ή κατά τη βούληση του προγραμματιστή.

3.13.5 Δυναμική ενημέρωση του πίνακα μέτρησης περασμάτων

Μία από τις απαιτήσεις του προβλήματος της επίλυσης του λαβυρίνθου ήταν και η δυναμική ενημέρωση του πίνακα μέτρησης των περασμάτων του κινητού πράκτορα και συγκεκριμένα του ιππότη. Στον Χάρτη της Τεχνητής Νοημοσύνης επιλέχθηκε η εύκολη λύση του Message Box και ένα String στο οποίο θα αποθηκεύονται τα δεδομένα. Δεν υπήρχε η απαίτηση της δυναμικής ενημέρωσης καθώς όταν επιλεγόταν να εμφανιστεί ο χάρτης, τότε η εφαρμογή σταματούσε για την περαιτέρω παρατήρηση του χάρτη.

Αντίθετα στον χάρτη περασμάτων η απαίτηση ήταν να υπάρχει δυναμική ενημέρωση ενώ θα μπορεί ο παρατηρητής να βλέπει και να τον μελετά. Όπως είναι φυσικό έπρεπε να ξεφύγουμε από την λύση των Message Boxes καθώς η νοοτροπία και ο σχεδιασμός τους δεν επέτρεπαν την πραγματοποίηση της συγκεκριμένης λειτουργίας (τουλάχιστον στον επιθυμητό βαθμό). Έτσι επιλέχθηκε η λύση της σχεδίασης μιας νέας φόρμας ειδικά για να χρησιμοποιηθεί για τον Χάρτη Περασμάτων. Όσον αφορά τα δεδομένα, χρησιμοποιούνται Text Boxes τα οποία είναι ισάριθμα με τον αριθμό των Frames του λαβυρίνθου. Επίσης, η διάταξή τους είναι τέτοια που να φαίνεται σχηματικά όπως ο λαβύρινθος και να μπορεί ο παρατηρητής εύκολα να συγκρίνει και να βλέπει τα περάσματα του κινητού πράκτορα. Βεβαίως το πρόβλημα δεν λύθηκε ακόμη καθώς έπρεπε να συνδεθούν τα Text Boxes με τα δεδομένα και να ανανεώνονται συνεχώς. Εδώ μπαίνει μια συνάρτηση, η countmap( ), η οποία κρατάει τα δεδομένα από τις μεταβλητές που αποθηκεύονται. Στη συνέχεια τα Text Boxes θεωρούνται αντικείμενα της φόρμας και συνδέονται άμεσα με την countmap( ). Ένα παράδειγμα είναι το παρακάτω:

countermap.Text1 = countmap(0)

Το countermap.Text1 είναι το πρώτο Text Box του λαβυρίνθου και το countmap(0) είναι ο αριθμός που δείχνει τον αριθμό περασμάτων από το πρώτο τετράγωνο του λαβυρίνθου. Το ίδιο συνεχίζεται για όλα τα υπόλοιπα Text Boxes της φόρμας. Ακόμη και στον μηδενισμό όλων των τιμών χρησιμοποιείται η ίδια φιλοσοφία, με την μόνη διαφορά αντί για τη συνάρτηση countmap( ), θέτουμε την τιμή μηδέν.

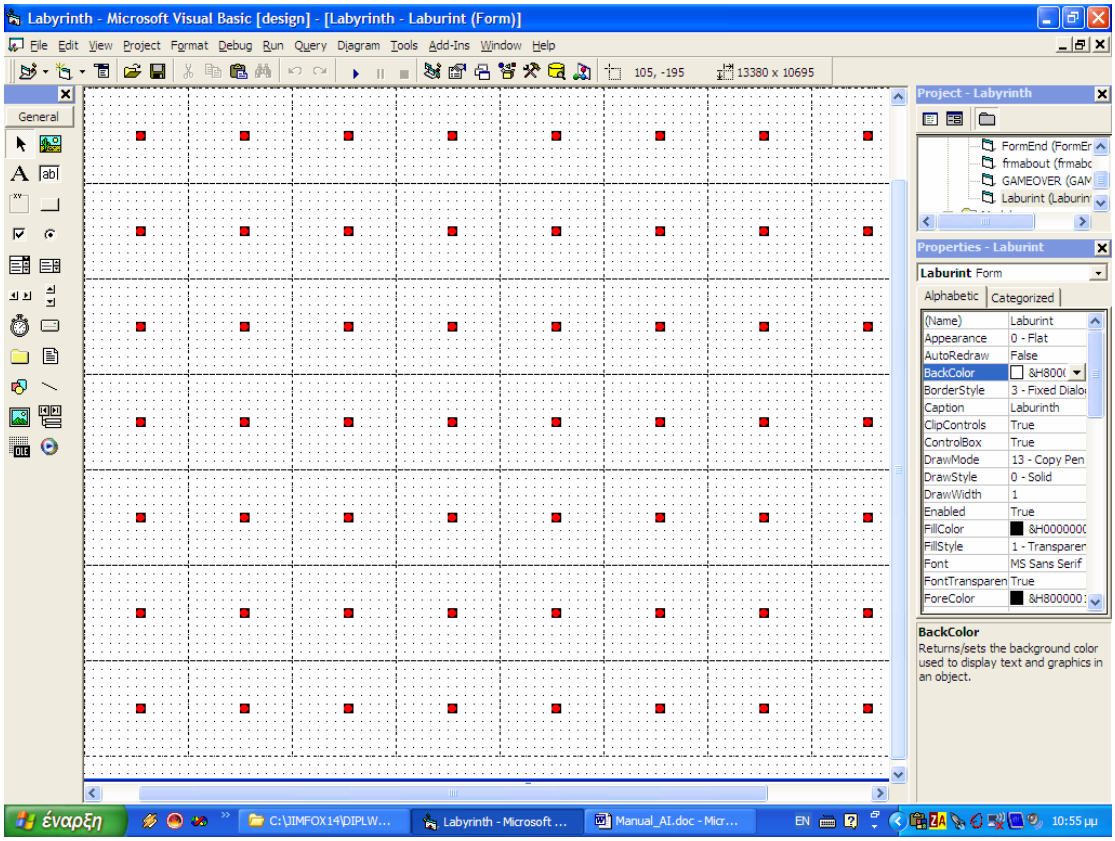
3.13.6 Εμφάνιση ίχνους

Ένα από τα προβλήματα που έπρεπε να λυθούν ήταν και η εμφάνιση ίχνους από τον ιππότη. Ο στόχος ήταν από όποιο τετράγωνο του λαβυρίνθου κι αν περνούσε ο ιππότης να φαίνεται και να το βλέπει ο παρατηρητής. Επίσης θα έπρεπε κατά την διάρκεια του μηδενισμού, όλα τα ίχνη που θα είχαν εμφανιστεί κατά τη διάρκεια της εφαρμογής, να εξαφανιστούν.

Καταρχήν, έπρεπε να βρεθεί το κατάλληλο εικονίδιο. Επιλέχθηκε ένα αρχείο τύπου ico αφού είναι συμβατό με την visual basic. Το χρώμα είναι κόκκινο για να μπορεί ο παρατηρητής να το βλέπει πιο άνετα, καθώς είναι έντονο και κάνει αντίθεση στο λαβύρινθο. Έπρεπε στη συνέχεια να φαίνεται το ίχνος σε κάθε frame της εφαρμογής. Η λύση ήταν απλή. Αντιγράφηκε το αρχικό εικονίδιο σε όλα τα frames και μπήκαν στο κέντρο τους. Έτσι σε κάθε frame του λαβυρίνθου μπήκε και ένα εικονίδιο και ευθυγραμμίστηκε στο κέντρο. Στην ουσία το κάθε εικονίδιο αποτελεί πλέον ένα αντικείμενο στη φόρμα. Έτσι για να εμφανιστεί το ίχνος όταν ο ιππότης περνούσε, ενεργοποιείται η ιδιότητα Visible του εικονιδίου και γίνεται true. Για παράδειγμα:

Image4.Visible = True

Το Image4 είναι το εικονίδιο, η Visible είναι η ιδιότητά του και true ή false είναι η κατάσταση στην οποία βρίσκεται. Παρακάτω φαίνεται επακριβώς πώς είναι τοποθετημένα τα εικονίδια στην εφαρμογή.



Το grid της εφαρμογής με τα εικονίδια τοποθετημένα.

3.14 Συμπέρασμα

Στο κεφάλαιο αυτό παρουσιάστηκε η υλοποίηση του προγράμματος επίλυσης του λαβυρίνθου με τεχνητή νοημοσύνη. Ο λαβύρινθος επιλύεται χωρίς καμία παρέμβαση από τον χρήστη, χρησιμοποιώντας κατάλληλους αλγορίθμους. Συγκεκριμένα στον λαβύρινθο βρίσκουν εφαρμογή οι αλγόριθμοι αναζήτησης και μάλιστα χρησιμοποιείται και ένας υβριδικός αλγόριθμος που κάνει χρήση μνήμης. Αναλύθηκε το interface της εφαρμογής, ο τρόπος με τον οποίο ο χρήστης μπορεί να βρει και να παρατηρήσει διάφορα στατιστικά στοιχεία επάνω στην αναζήτηση για τη λύση, αναλύθηκε ο σχεδιασμός και ο κώδικας της εφαρμογής με τέτοιον τρόπο ώστε να γίνει κατανοητή η φιλοσοφία του προγραμματισμού της εφαρμογής. Αναφερθήκαμε αναλυτικά στις βασικές συναρτήσεις της εφαρμογής καθώς και στον

κώδικά τους. Οι συναρτήσεις αυτές περιλαμβάνουν μετρητές κινήσεων, εμφάνιση στατιστικών δεδομένων στην οθόνη, καθώς και συναρτήσεις που εμφανίζουν κάποιο ίχνος στον λαβύρινθο για να φαίνεται η κίνηση του κινητού πράκτορα. Τέλος αναφερόμαστε σε κάποια από τα προβλήματα που αντιμετωπίστηκαν και είναι άξια αναφοράς καθώς επεξηγούν καλύτερα τον τρόπο που δημιουργήθηκε η εφαρμογή.

ΚΕΦΑΛΑΙΟ 4

ΣΥΜΠΕΡΑΣΜΑΤΑ – ΠΡΟΟΠΤΙΚΕΣ

Στόχος της εργασίας είναι η επίλυση ενός τυχαίου λαβυρίνθου χρησιμοποιώντας έναν κινητό πράκτορα ο οποίος θα κάνει χρήση τεχνητής νοημοσύνης. Ο λαβύρινθος θα δημιουργείται τυχαία έτσι ώστε η μορφή του να μην είναι σταθερή και ο πράκτορας θα πρέπει να έχει την δυνατότητα να επιλύει όλους τους δυνατούς λαβύρινθους, αν φυσικά λύνονται. Για να αυξηθεί η δυσκολία προστέθηκε και ένα κινητό εμπόδιο το οποίο κινείται στον λαβύρινθο με σκοπό να ανακόψει την πορεία του κινητού πράκτορα.

Όπως είναι φυσικό ο κινητός πράκτορας χρησιμοποιεί κάποιον αλγόριθμο για να μπορέσει να κινηθεί και να φτάσει στη λύση. Ο αλγόριθμος αυτός είναι ένας υβριδικός αλγόριθμος αναζήτησης ο οποίος κάνει χρήση μνήμης. Αποτελεί ένα συνδυασμό μεταξύ του αλγορίθμου αναζήτησης σε βάθος και ενός αλγορίθμου ευρετικής αναζήτησης κάνοντας χρήση μνήμης. Έτσι μπορεί να κινηθεί στο λαβύρινθο γνωρίζοντας από που έχει περάσει ήδη. Η εφαρμογή περιλαμβάνει και κάποια επιπλέον χαρακτηριστικά όπως για παράδειγμα τον χάρτη που συμπληρώνεται από την Τ.Ν. και είναι δυναμικός, δηλαδή όσο προχωρά ο πράκτορας ενημερώνεται ταυτόχρονα και ο χάρτης, στατιστικά στοιχεία που μας δείχνουν πόσα βήματα χρειάστηκε ο παίκτης για να φτάσει στον τερματισμό. Επίσης υπάρχει η δυνατότητα επανεκκίνησης του λαβυρίνθου είτε αρχικοποιώντας τις μεταβλητές και μηδενίζοντας τις αξίες τους, είτε απλά δημιουργώντας έναν νέο λαβύρινθο (τυχαίο).

Στατιστικά στην συγκεκριμένη εφαρμογή γύρω στο 95% των λαβυρίνθων, μπορεί να φτάσει σε κάποια λύση, είτε είναι εύρεση της εξόδου του λαβυρίνθου, είτε είναι game over, δηλαδή το εμπόδιο να έχει βρει τον παίκτη και να έχει ανακόψει την πορεία του. Από αυτό το σύνολο των φορών που έχει βρεθεί λύση στον λαβύρινθο και έχει ολοκληρωθεί η αναζήτηση, το 45% ο ιππότης θα βρει την έξοδο, ενώ το 55% το κινητό εμπόδιο θα ανακόψει την πορεία του παίκτη.

Η επίλυση του λαβυρίνθου μπορεί να μεταφερθεί και σε άλλα παρόμοια προβλήματα και ιδίως στον πραγματικό κόσμο. Στα δίκτυα των ηλεκτρονικών υπολογιστών για παράδειγμα μπορεί ο υβριδικός αλγόριθμος, που δημιουργήθηκε για τις ανάγκες της συγκεκριμένης εργασίας, να βρει εφαρμογή. Πιο συγκεκριμένα στη μεταφορά αρχείων μέσω διαδικτύου και μάλιστα αρχείων πολυμέσων. Αυτό γιατί τα δίκτυα είναι περίπλοκα σαν τον λαβύρινθο, αλλά ταυτόχρονα για ελάχιστο χρονικό διάστημα παραμένουν σταθερά. Αλλάζει συνεχώς η μορφή και το κόστος μεταφοράς

των αρχείων. Διατηρείται δηλαδή ο τυχαίος χαρακτήρας που υπάρχει και στον λαβύρινθο. Επίσης σε συνδυασμό με την ρομποτική ο αλγόριθμος που έχουμε δημιουργήσει μπορεί να εφαρμοστεί στην αυτόνομη κίνηση των αυτοκινήτων η οποία τα τελευταία χρόνια έχει κάνει πολύ μεγάλες προόδους ιδιαίτερα στην Αμερική με τη DARPA. Ακόμη και στα προβλήματα παιγνίων όπως στο σκάκι, στο puzzle με τους αριθμούς και στο γνωστό σε όλους μας pac-man υπάρχουν αλγόριθμοι (έστω κατ' ανάγκη τροποποιημένοι) που βασίζονται στους αλγόριθμους αναζήτησης.

Σήμερα παρά όλη την εξέλιξη που υπάρχει στον τομέα των ηλεκτρονικών υπολογιστών δεν έχουμε φτάσει στο βέλτιστο σημείο. Η εξέλιξη είναι συνεχής και τίποτα δεν παραμένει στάσιμο για πολύ καιρό. Έτσι και η εφαρμογή που δημιουργήσαμε για την συγκεκριμένη εργασία μπορεί να βελτιωθεί. Ο υβριδικός αλγόριθμος αναζήτησης που χρησιμοποιεί ο κινητός πράκτορας μπορεί να βελτιωθεί παίρνοντας τα κύρια χαρακτηριστικά του A*, ενός από τους πιο δημοφιλείς και εξελιγμένους αλγόριθμους αναζήτησης. Έτσι θα γίνεται χρήση ανοικτού και κλειστού καταλόγου και θα καταχωρούνται όλοι οι κόμβοι που προσπελαύνει ο κινητός πράκτορας έτσι ώστε να μπορεί να ξέρει ανά πάσα στιγμή ποια είναι η βέλτιστη διαδρομή. Εισάγοντας και τον παράγοντα κόστος θα μπορεί να συγκρίνει πιθανές διαδρομές και να επιλέγει την καλύτερη και την γρηγορότερη.

Η τεχνητή νοημοσύνη θεωρείται από πολλούς ότι θα βοηθήσει σημαντικά στην επίλυση των προβλημάτων που συνδέονται με την έρευνα και την ανάπτυξη και θα συντελέσει στη βελτίωση της ποιότητας ζωής σε ατομικό αλλά και σε κοινωνικό επίπεδο. Ωστόσο υπάρχουν φόβοι για αρνητικές συνέπειες στη ζωή του ανθρώπου όπως η αύξηση της ανεργίας, η δυσκολία ελέγχου των συστημάτων παραγωγής αφού θα αποκτήσουν αυτονομία λειτουργίας. βέβαια όλες αυτές οι αλλαγές θα γίνουν με αργό ρυθμό και πολλοί είναι αυτοί που σκέφτονται θετικά και πιστεύουν ότι οι αλλαγές αυτές θα ενσωματωθούν στο κοινωνικό σύνολο χωρίς να υπάρχουν ιδιαίτερα προβλήματα. Φυσικά κανείς δεν μπορεί να μιλήσει με σιγουριά, γι' αυτό και ο χρόνος είναι αυτός που θα δείξει το αν έχουν δίκιο οι σκεπτικιστές ή οι αισιόδοξοι....

ΠΑΡΑΡΤΗΜΑ

```
Dim countL As Integer   ' Μεταβλητή μετρητής
Dim nisse As Integer    ' Όρισμα για κουτάκι
Dim Map(55) As Byte     ' Χάρτης
Dim AImap(55) As Byte   ' Χάρτης τεχνητής νοημοσύνης
Dim pcpos As Byte       ' Θέση χαρακτήρα
Dim pcend As Byte       ' Θέση τέλους
Dim vroc As Integer  'μετρητης επαναλήψεων
Dim pcwumpus As Byte  'Θέση wumpus
Dim o As Integer
Public TimerClock  As Double          ' Clock για pause
Public OKtoContinue As Boolean
Dim steps As Integer 'μετρητής βημάτων μέχρι το τέλος για τον player
Dim stepsw As Integer 'μετρητής βημάτων μέχρι το τέλος για το wumpus
Dim totalsteps As Integer 'συνολικά βήματα μέχρι το τέλος
Dim countmap(55) As Integer 'Μετρά τις φορές ελέγχου στα τετράγωνα του
χάρτη
Option Explicit


Private Sub Command1_Click()
steps = 0
stepsw = 0
totalsteps = 0
While (pcpos <> pcend)
vroc = 0
'player επανω
For vroc = 0 To 10
lookleft
lookright
If pcpos > 7 Then
   If Map(pcpos) <> 3 Then
      If (Map(pcpos - 8) <> 5) Then
      If (pcpos - 8 <> pcwumpus) Then
      If countmap(pcpos - 8) < 25 Then
pcpos = pcpos - 8
steps = steps + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
            counttimes
lookleft
lookright
            showcounter
            iconmap
            checkup
            Pause (0.15)
CheckEnd
CheckKill
trace
```

```
End If
End If
End If
End If
End If
Next vroc

'wumpus δεξια

For vroc = 0 To 10
lookforplayer

If (pcwumpus <> 7 And pcwumpus <> 15 And pcwumpus <> 23 And
pcwumpus <> 31 And pcwumpus <> 39 And pcwumpus <> 47 And pcwumpus
<> 55) Then
   If (Map(pcwumpus) <> 4 And Map(pcwumpus + 1) <> 2) Then

      If pcwumpus > 7 Then
            If Map(pcwumpus - 8) = 5 Or (Map(pcwumpus) = 3) Then
             pcwumpus = pcwumpus + 1
             stepsw = stepsw + 1
            Wumpus.Top = Image1(pcwumpus).Top +
Image1(pcwumpus).Height / 2 - Wumpus.Height / 2
            Wumpus.Left = Image1(pcwumpus).Left +
Image1(pcwumpus).Width / 2 - Wumpus.Width / 2

Pause (0.15)


showcounter
CheckEnd
CheckKill
trace
End If
End If
End If
End If



If (pcwumpus <> 7 And pcwumpus <> 15 And pcwumpus <> 23 And
pcwumpus <> 31 And pcwumpus <> 39 And pcwumpus <> 47 And pcwumpus
<> 55) Then
   If (Map(pcwumpus) <> 4 And Map(pcwumpus + 1) <> 2) Then
     If pcwumpus < 7 Then
 pcwumpus = pcwumpus + 1
 stepsw = stepsw + 1
            Wumpus.Top = Image1(pcwumpus).Top +
Image1(pcwumpus).Height / 2 - Wumpus.Height / 2
```

```
            Wumpus.Left = Image1(pcwumpus).Left +
Image1(pcwumpus).Width / 2 - Wumpus.Width / 2

Pause (0.15)

showcounter
CheckEnd
CheckKill
trace
End If
End If
End If
Next vroc

'wumpus επάνω

 For vroc = 0 To 10
 lookforplayer

If pcwumpus > 7 Then
   If Map(pcwumpus) <> 3 Then
      If (Map(pcwumpus - 8) <> 5) Then
pcwumpus = pcwumpus - 8
stepsw = stepsw + 1
            Wumpus.Top = Image1(pcwumpus).Top +
Image1(pcwumpus).Height / 2 - Wumpus.Height / 2
            Wumpus.Left = Image1(pcwumpus).Left +
Image1(pcwumpus).Width / 2 - Wumpus.Width / 2
Pause (0.15)


      showcounter
            CheckEnd
CheckKill
trace
End If
End If
End If
Next vroc



'player δεξιά
For vroc = 0 To 10
If (pcpos <> 7 And pcpos <> 15 And pcpos <> 23 And pcpos <> 31 And
pcpos <> 39 And pcpos <> 47 And pcpos <> 55) Then
   If (Map(pcpos) <> 4 And Map(pcpos + 1) <> 2) Then

      If pcpos > 7 Then
            If Map(pcpos - 8) = 5 Or (Map(pcpos) = 3) Then
```

```
            If (pcpos + 1 <> pcwumpus) Then
            If countmap(pcpos + 1) < 25 Then
             pcpos = pcpos + 1
             steps = steps + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
counttimes
lookforplayer
showcounter
iconmap
pclookup
            checkright
            Pause (0.15)

            CheckEnd
CheckKill
trace
End If
End If
End If
End If
End If
End If


If (pcpos <> 7 And pcpos <> 15 And pcpos <> 23 And pcpos <> 31 And
pcpos <> 39 And pcpos <> 47 And pcpos <> 55) Then
   If (Map(pcpos) <> 4 And Map(pcpos + 1) <> 2) Then
     If pcpos < 7 Then
     If (pcpos + 1 <> pcwumpus) Then
     If countmap(pcpos + 1) < 25 Then
 pcpos = pcpos + 1
 steps = steps + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
counttimes
lookforplayer
         showcounter
      iconmap
            Pause (0.15)
            checkright
            CheckEnd

CheckKill
trace
End If
```

```
End If
End If
End If
End If
Next vroc


'wumpus αριστερά


For vroc = 0 To 10
lookforplayer

If (pcwumpus <> 0 And pcwumpus <> 8 And pcwumpus <> 16 And
pcwumpus <> 24 And pcwumpus <> 32 And pcwumpus <> 40 And pcwumpus
<> 48) Or (pcwumpus = 7 Or pcwumpus = 15 Or pcwumpus = 23 Or
pcwumpus = 31 Or pcwumpus = 39 Or pcwumpus = 47 Or pcwumpus = 55)
Then
  If Map(pcwumpus) <> 2 Then
  If Map(pcwumpus - 1) <> 4 Then
  pcwumpus = pcwumpus - 1
  stepsw = stepsw + 1
            Wumpus.Top = Image1(pcwumpus).Top +
Image1(pcwumpus).Height / 2 - Wumpus.Height / 2
            Wumpus.Left = Image1(pcwumpus).Left +
Image1(pcwumpus).Width / 2 - Wumpus.Width / 2
          Pause (0.15)

            showcounter
            CheckEnd
CheckKill
trace
End If
End If
End If
Next vroc

'player πάνω
For vroc = 0 To 5
lookleft
lookright
If pcpos > 7 Then
   If Map(pcpos) <> 3 Then
      If (Map(pcpos - 8) <> 5) Then
      If (pcpos - 8 <> pcwumpus) Then
      If countmap(pcpos - 8) < 25 Then
pcpos = pcpos - 8
steps = steps + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
```

```
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
counttimes
showcounter
iconmap
       lookleft
        lookright
            checkup
            Pause (0.15)

            CheckEnd
CheckKill
trace
End If
End If
End If
End If
End If
Next vroc


 'wumpus πάνω

 For vroc = 0 To 10
 lookforplayer
If pcwumpus > 7 Then
   If Map(pcwumpus) <> 3 Then
      If (Map(pcwumpus - 8) <> 5) Then
pcwumpus = pcwumpus - 8
stepsw = stepsw + 1
            Wumpus.Top = Image1(pcwumpus).Top +
Image1(pcwumpus).Height / 2 - Wumpus.Height / 2
            Wumpus.Left = Image1(pcwumpus).Left +
Image1(pcwumpus).Width / 2 - Wumpus.Width / 2
           Pause (0.15)
         showcounter
    iconmap
            CheckEnd
CheckKill
trace
End If
End If
End If
Next vroc



'player αριστερά

For vroc = 0 To 6
```

```
If (pcpos <> 0 And pcpos <> 8 And pcpos <> 16 And pcpos <> 24 And pcpos
<> 32 And pcpos <> 40 And pcpos <> 48) Or (pcpos = 7 Or pcpos = 15 Or
pcpos = 23 Or pcpos = 31 Or pcpos = 39 Or pcpos = 47 Or pcpos = 55) Then
  If Map(pcpos) <> 2 Then
  If Map(pcpos - 1) <> 4 Then
  If (pcpos - 1 <> pcwumpus) Then
  If countmap(pcpos - 1) < 25 Then
  pcpos = pcpos - 1
  steps = steps + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
counttimes
showcounter
iconmap
      pclookup
            checkleft
            Pause (0.15)

            CheckEnd
CheckKill
trace
End If
End If
End If
End If
End If
Next vroc

'wumpus δεξιά

For vroc = 0 To 10
lookforplayer

If (pcwumpus <> 7 And pcwumpus <> 15 And pcwumpus <> 23 And
pcwumpus <> 31 And pcwumpus <> 39 And pcwumpus <> 47 And pcwumpus
<> 55) Then
   If (Map(pcwumpus) <> 4 And Map(pcwumpus + 1) <> 2) Then

      If pcwumpus > 7 Then
            If Map(pcwumpus - 8) = 5 Or (Map(pcwumpus) = 3) Then
             pcwumpus = pcwumpus + 1
             stepsw = stepsw + 1
            Wumpus.Top = Image1(pcwumpus).Top +
Image1(pcwumpus).Height / 2 - Wumpus.Height / 2
            Wumpus.Left = Image1(pcwumpus).Left +
Image1(pcwumpus).Width / 2 - Wumpus.Width / 2
```

```
Pause (0.15)
        showcounter


            CheckEnd
CheckKill
trace
End If
End If
End If
End If



If (pcwumpus <> 7 And pcwumpus <> 15 And pcwumpus <> 23 And
pcwumpus <> 31 And pcwumpus <> 39 And pcwumpus <> 47 And pcwumpus
<> 55) Then
   If (Map(pcwumpus) <> 4 And Map(pcwumpus + 1) <> 2) Then
     If pcwumpus < 7 Then
 pcwumpus = pcwumpus + 1
 stepsw = stepsw + 1
            Wumpus.Top = Image1(pcwumpus).Top +
Image1(pcwumpus).Height / 2 - Wumpus.Height / 2
            Wumpus.Left = Image1(pcwumpus).Left +
Image1(pcwumpus).Width / 2 - Wumpus.Width / 2
counttimes
Pause (0.15)


            showcounter
            CheckEnd
CheckKill
trace
End If
End If
End If
Next vroc


'player κάτω
For vroc = 0 To 2

If pcpos < 48 Then
If (Map(pcpos) <> 5 And Map(pcpos + 8) <> 3) Then
If (pcpos + 8 <> pcwumpus) Then
If countmap(pcpos + 8) < 25 Then
pcpos = pcpos + 8
steps = steps + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
```

```
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
counttimes
      iconmap
            checkdown
            Pause (0.15)
            showcounter
            CheckEnd
CheckKill
trace
End If
End If
End If
End If
If (pcpos <> 7 And pcpos <> 15 And pcpos <> 23 And pcpos <> 31 And
pcpos <> 39 And pcpos <> 47 And pcpos <> 55) Then
   If (Map(pcpos) <> 4 And Map(pcpos + 1) <> 2) Then

       If pcpos > 7 Then
            If (pcpos + 1 <> pcwumpus) Then
            If countmap(pcpos + 1) < 25 Then
             pcpos = pcpos + 1
             steps = steps + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
counttimes
showcounter
iconmap
            checkright
            Pause (0.15)

            CheckEnd
CheckKill
trace
End If
End If
End If
End If
End If
Next vroc

'wumpus κάτω

For vroc = 0 To 10
lookforplayer

If pcwumpus < 48 Then
If (Map(pcwumpus) <> 5 And Map(pcwumpus + 8) <> 3) Then
```

```
pcwumpus = pcwumpus + 8
stepsw = stepsw + 1
            Wumpus.Top = Image1(pcwumpus).Top +
Image1(pcwumpus).Height / 2 - Wumpus.Height / 2
            Wumpus.Left = Image1(pcwumpus).Left +
Image1(pcwumpus).Width / 2 - Wumpus.Width / 2
Pause (0.15)
lookforplayer
pclookup
         showcounter


            CheckEnd
CheckKill
trace
End If
End If
If (pcwumpus <> 7 And pcwumpus <> 15 And pcwumpus <> 23 And
pcwumpus <> 31 And pcwumpus <> 39 And pcwumpus <> 47 And pcwumpus
<> 55) Then
   If (Map(pcwumpus) <> 4 And Map(pcwumpus + 1) <> 2) Then

      If pcwumpus > 7 Then
            pcwumpus = pcwumpus + 1
            stepsw = stepsw + 1
            Wumpus.Top = Image1(pcwumpus).Top +
Image1(pcwumpus).Height / 2 - Wumpus.Height / 2
            Wumpus.Left = Image1(pcwumpus).Left +
Image1(pcwumpus).Width / 2 - Wumpus.Width / 2
counttimes
Pause (0.15)
            showcounter


            CheckEnd
CheckKill
trace
End If
End If
End If
Next vroc


'player πάνω
For vroc = 0 To 10
lookleft
lookright
If pcpos > 7 Then
   If Map(pcpos) <> 3 Then
      If (Map(pcpos - 8) <> 5) Then
```

```
      If (pcpos - 8 <> pcwumpus) Then
      If countmap(pcpos - 8) < 25 Then
pcpos = pcpos - 8
steps = steps + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
counttimes
lookleft
lookright
showcounter
iconmap
            checkright
            Pause (0.15)
CheckEnd
CheckKill
trace
End If
End If
End If
End If
End If
Next vroc

'wumpus αριστερά

For vroc = 0 To 10
lookforplayer

If (pcwumpus <> 0 And pcwumpus <> 8 And pcwumpus <> 16 And
pcwumpus <> 24 And pcwumpus <> 32 And pcwumpus <> 40 And pcwumpus
<> 48) Or (pcwumpus = 7 Or pcwumpus = 15 Or pcwumpus = 23 Or
pcwumpus = 31 Or pcwumpus = 39 Or pcwumpus = 47 Or pcwumpus = 55)
Then
  If Map(pcwumpus) <> 2 Then
  If Map(pcwumpus - 1) <> 4 Then
  pcwumpus = pcwumpus - 1
  stepsw = stepsw + 1
            Wumpus.Top = Image1(pcwumpus).Top +
Image1(pcwumpus).Height / 2 - Wumpus.Height / 2
            Wumpus.Left = Image1(pcwumpus).Left +
Image1(pcwumpus).Width / 2 - Wumpus.Width / 2
Pause (0.15)
lookforplayer
        showcounter


            CheckEnd
CheckKill
```

```
trace
End If
End If
End If
Next vroc

'player αριστερά
For vroc = 0 To 5

If (pcpos <> 0 And pcpos <> 8 And pcpos <> 16 And pcpos <> 24 And pcpos
<> 32 And pcpos <> 40 And pcpos <> 48) Or (pcpos = 7 Or pcpos = 15 Or
pcpos = 23 Or pcpos = 31 Or pcpos = 39 Or pcpos = 47 Or pcpos = 55 Or
Map(pcpos) = 4 Or Map(pcpos + 1) = 2) Then
  If Map(pcpos) <> 2 Then
  If Map(pcpos - 1) <> 4 Then
  If (pcpos - 1 <> pcwumpus) Then
  If countmap(pcpos - 1) < 25 Then
    pcpos = pcpos - 1
    steps = steps + 1
          PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
          PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
counttimes
showcounter
     iconmap
      pclookup
          checkleft
          Pause (0.15)

          CheckEnd
CheckKill
trace
End If
End If
End If
End If
End If
Next vroc

'wumpus πάνω

 For vroc = 0 To 10
lookforplayer

If pcwumpus > 7 Then
   If Map(pcwumpus) <> 3 Then
      If (Map(pcwumpus - 8) <> 5) Then
pcwumpus = pcwumpus - 8
stepsw = stepsw + 1
```

```
            Wumpus.Top = Image1(pcwumpus).Top +
Image1(pcwumpus).Height / 2 - Wumpus.Height / 2
            Wumpus.Left = Image1(pcwumpus).Left +
Image1(pcwumpus).Width / 2 - Wumpus.Width / 2
Pause (0.15)
         showcounter


            CheckEnd
CheckKill
trace
End If
End If
End If
Next vroc



'player δεξιά
For vroc = 0 To 5

If (pcpos <> 7 And pcpos <> 15 And pcpos <> 23 And pcpos <> 31 And
pcpos <> 39 And pcpos <> 47 And pcpos <> 55) Then
   If (Map(pcpos) <> 4 And Map(pcpos + 1) <> 2) Then

         If pcpos > 7 Then
            If Map(pcpos - 8) = 5 Or (Map(pcpos) = 3) Then
            If (pcpos + 1 <> pcwumpus) Then
            If countmap(pcpos + 1) < 25 Then
         pcpos = pcpos + 1
         steps = steps + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
counttimes
            pclookup
            checkright
            Pause (0.15)
            showcounter
          iconmap
            CheckEnd
CheckKill
trace
End If
End If
End If
End If
End If
```

```
If (pcpos <> 7 And pcpos <> 15 And pcpos <> 23 And pcpos <> 31 And
pcpos <> 39 And pcpos <> 47 And pcpos <> 55) Then
    If (Map(pcpos) <> 4 And Map(pcpos + 1) <> 2) Then
        If (pcpos < 7) Then
        If countmap(pcpos + 1) < 25 Then
 pcpos = pcpos + 1
 steps = steps + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
counttimes
pclookup
            checkright
            Pause (0.15)
            showcounter
            iconmap
            CheckEnd
CheckKill
trace
End If
End If
End If
End If
End If
Next vroc


'wumpus αριστερά

For vroc = 0 To 10
lookforplayer

If (pcwumpus <> 0 And pcwumpus <> 8 And pcwumpus <> 16 And
pcwumpus <> 24 And pcwumpus <> 32 And pcwumpus <> 40 And pcwumpus
<> 48) Or (pcwumpus = 7 Or pcwumpus = 15 Or pcwumpus = 23 Or
pcwumpus = 31 Or pcwumpus = 39 Or pcwumpus = 47 Or pcwumpus = 55)
Then
  If Map(pcwumpus) <> 2 Then
  If Map(pcwumpus - 1) <> 4 Then
  pcwumpus = pcwumpus - 1
  stepsw = stepsw + 1
            Wumpus.Top = Image1(pcwumpus).Top +
Image1(pcwumpus).Height / 2 - Wumpus.Height / 2
            Wumpus.Left = Image1(pcwumpus).Left +
Image1(pcwumpus).Width / 2 - Wumpus.Width / 2

       Pause (0.15)
```

```
            showcounter
            CheckEnd
CheckKill
trace
End If
End If
End If
Next vroc



'player κάτω
For vroc = 0 To 3
If pcpos < 48 Then
If (Map(pcpos) <> 5 And Map(pcpos + 8) <> 3) Then
If (pcpos + 8 <> pcwumpus) Then
pcpos = pcpos + 8
steps = steps + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
counttimes
            checkdown
            Pause (0.15)
            showcounter
            iconmap
            CheckEnd
CheckKill
trace
End If
End If
End If
If (pcpos <> 0 And pcpos <> 8 And pcpos <> 16 And pcpos <> 24 And pcpos
<> 32 And pcpos <> 40 And pcpos <> 48) Or (pcpos = 7 Or pcpos = 15 Or
pcpos = 23 Or pcpos = 31 Or pcpos = 39 Or pcpos = 47 Or pcpos = 55 Or
Map(pcpos) = 4 Or Map(pcpos + 1) = 2) Then
  If Map(pcpos) <> 2 Then
  If Map(pcpos - 1) <> 4 Then
  If pcpos - 1 <> pcwumpus Then
    pcpos = pcpos - 1
    steps = steps + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
counttimes
            checkleft
            Pause (0.15)
            showcounter
```

```
            iconmap
            CheckEnd
CheckKill
trace
End If
End If
End If
End If
Next vroc

'wumpus πάνω

 For vroc = 0 To 10
lookforplayer

If pcwumpus > 7 Then
   If Map(pcwumpus) <> 3 Then
      If (Map(pcwumpus - 8) <> 5) Then
pcwumpus = pcwumpus - 8
stepsw = stepsw + 1
            Wumpus.Top = Image1(pcwumpus).Top +
Image1(pcwumpus).Height / 2 - Wumpus.Height / 2
            Wumpus.Left = Image1(pcwumpus).Left +
Image1(pcwumpus).Width / 2 - Wumpus.Width / 2
Pause (0.15)


            showcounter
            CheckEnd
CheckKill
trace
End If
End If
End If
Next vroc

'player αριστερά

For vroc = 0 To 10

If (pcpos <> 0 And pcpos <> 8 And pcpos <> 16 And pcpos <> 24 And pcpos
<> 32 And pcpos <> 40 And pcpos <> 48) Or (pcpos = 7 Or pcpos = 15 Or
pcpos = 23 Or pcpos = 31 Or pcpos = 39 Or pcpos = 47 Or pcpos = 55) Then
  If Map(pcpos) <> 2 Then
  If Map(pcpos - 1) <> 4 Then
  If pcpos - 1 <> pcwumpus Then
  pcpos = pcpos - 1
  steps = steps + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
```

```
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
counttimes
      pclookup
            checkleft
            Pause (0.15)
            showcounter
            iconmap
            CheckEnd
CheckKill
trace
End If
End If
End If
End If
Next vroc


'player πάνω
For vroc = 0 To 10
lookleft
lookright
If pcpos > 7 Then
   If Map(pcpos) <> 3 Then
      If (Map(pcpos - 8) <> 5) Then
      If pcpos - 8 <> pcwumpus Then
pcpos = pcpos - 8
steps = steps + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
counttimes
      lookright
      lookleft
            showcounter
            iconmap
            Pause (0.15)
            checkup
            CheckEnd
CheckKill
trace
End If
End If
End If
End If
Next vroc

'wumpus δεξιά
```

```
For vroc = 0 To 10
lookforplayer

If (pcwumpus <> 0 And pcwumpus <> 8 And pcwumpus <> 16 And
pcwumpus <> 24 And pcwumpus <> 32 And pcwumpus <> 40 And pcwumpus
<> 48) Or (pcwumpus = 7 Or pcwumpus = 15 Or pcwumpus = 23 Or
pcwumpus = 31 Or pcwumpus = 39 Or pcwumpus = 47 Or pcwumpus = 55)
Then
  If Map(pcwumpus) <> 2 Then
  If Map(pcwumpus - 1) <> 4 Then
  pcwumpus = pcwumpus - 1
  stepsw = stepsw + 1
            Wumpus.Top = Image1(pcwumpus).Top +
Image1(pcwumpus).Height / 2 - Wumpus.Height / 2
            Wumpus.Left = Image1(pcwumpus).Left +
Image1(pcwumpus).Width / 2 - Wumpus.Width / 2
Pause (0.15)
            showcounter

            iconmap
            CheckEnd
CheckKill
trace
End If
End If
End If
Next vroc



'player αριστερά
For vroc = 0 To 10

If (pcpos <> 0 And pcpos <> 8 And pcpos <> 16 And pcpos <> 24 And pcpos
<> 32 And pcpos <> 40 And pcpos <> 48) Or (pcpos = 7 Or pcpos = 15 Or
pcpos = 23 Or pcpos = 31 Or pcpos = 39 Or pcpos = 47 Or pcpos = 55 Or
Map(pcpos) = 4 Or Map(pcpos + 1) = 2) Then
  If Map(pcpos) <> 2 Then
     If Map(pcpos - 1) <> 4 Then

     If pcpos < 48 Then
        If Map(pcpos + 8) = 3 Or Map(pcpos) = 5 Then
        If pcpos - 1 <> pcwumpus Then
  pcpos = pcpos - 1
  steps = steps + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
counttimes
```

```
            pclookup
            checkleft
            Pause (0.15)
            showcounter
            iconmap
            CheckEnd
CheckKill
trace
End If
End If
End If
End If
End If
End If

If (pcpos <> 0 And pcpos <> 8 And pcpos <> 16 And pcpos <> 24 And pcpos
<> 32 And pcpos <> 40 And pcpos <> 48) Or (pcpos = 7 Or pcpos = 15 Or
pcpos = 23 Or pcpos = 31 Or pcpos = 39 Or pcpos = 47 Or pcpos = 55) Then
  If Map(pcpos) <> 2 Then
  If Map(pcpos - 1) <> 4 Then
    If pcpos > 48 Then
    If pcpos - 1 <> pcwumpus Then
  pcpos = pcpos - 1
  steps = steps + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
counttimes
pclookup
            checkleft
            Pause (0.15)
            showcounter
            iconmap
            CheckEnd
CheckKill
trace
End If
End If
End If
End If
End If
Next vroc


Wend

End Sub
```

```
Private Sub exit_Click()
End
End Sub

Private Sub mnuAbout_Click()
   frmabout.Show
End Sub

Private Sub mnushowcountingmap_Click()

countermap.Visible = True

End Sub

' Εμφάνιση χάρτη ΤΝ
Private Sub mnuShowMap_Click()
AI_IconMap.Visible = True
End Sub

Private Sub iconmap()
If AI_IconMap.Visible = True Then
AI_IconMap.Text1 = AImap(0)
AI_IconMap.Text8 = AImap(1)
AI_IconMap.Text15 = AImap(2)
AI_IconMap.Text22 = AImap(3)
AI_IconMap.Text29 = AImap(4)
AI_IconMap.Text36 = AImap(5)
AI_IconMap.Text43 = AImap(6)
AI_IconMap.Text50 = AImap(7)
AI_IconMap.Text2 = AImap(8)
AI_IconMap.Text9 = AImap(9)
AI_IconMap.Text16 = AImap(10)
AI_IconMap.Text23 = AImap(11)
AI_IconMap.Text30 = AImap(12)
AI_IconMap.Text37 = AImap(13)
AI_IconMap.Text44 = AImap(14)
AI_IconMap.Text51 = AImap(15)
AI_IconMap.Text3 = AImap(16)
AI_IconMap.Text10 = AImap(17)
AI_IconMap.Text17 = AImap(18)
AI_IconMap.Text24 = AImap(19)
AI_IconMap.Text31 = AImap(20)
AI_IconMap.Text38 = AImap(21)
AI_IconMap.Text45 = AImap(22)
AI_IconMap.Text52 = AImap(23)
AI_IconMap.Text4 = AImap(24)
```

```
AI_IconMap.Text11 = AImap(25)
AI_IconMap.Text18 = AImap(26)
AI_IconMap.Text25 = AImap(27)
AI_IconMap.Text32 = AImap(28)
AI_IconMap.Text39 = AImap(29)
AI_IconMap.Text46 = AImap(30)
AI_IconMap.Text53 = AImap(31)
AI_IconMap.Text5 = AImap(32)
AI_IconMap.Text12 = AImap(33)
AI_IconMap.Text19 = AImap(34)
AI_IconMap.Text26 = AImap(35)
AI_IconMap.Text33 = AImap(36)
AI_IconMap.Text40 = AImap(37)
AI_IconMap.Text47 = AImap(38)
AI_IconMap.Text54 = AImap(39)
AI_IconMap.Text6 = AImap(40)
AI_IconMap.Text13 = AImap(41)
AI_IconMap.Text20 = AImap(42)
AI_IconMap.Text27 = AImap(43)
AI_IconMap.Text34 = AImap(44)
AI_IconMap.Text40 = AImap(45)
AI_IconMap.Text47 = AImap(46)
AI_IconMap.Text54 = AImap(47)
AI_IconMap.Text7 = AImap(48)
AI_IconMap.Text14 = AImap(49)
AI_IconMap.Text21 = AImap(50)
AI_IconMap.Text28 = AImap(51)
AI_IconMap.Text35 = AImap(52)
AI_IconMap.Text42 = AImap(53)
AI_IconMap.Text49 = AImap(54)
AI_IconMap.Text56 = AImap(55)
End If
End Sub


' Κουμπί για κάτω
Private Sub cmdDown_Click()
   If pcpos < 48 Then
      If Map(pcpos) <> 5 Then
         If Map(pcpos + 8) <> 3 Then
            pcpos = pcpos + 8
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
         End If
      End If
   End If
   CheckEnd
End Sub
```

```
' Κουμπί για αριστερά
Private Sub cmdLeft_Click()
   If pcpos <> 0 And pcpos <> 8 And pcpos <> 16 And pcpos <> 24 And
pcpos <> 32 And pcpos <> 40 And pcpos <> 48 Then
      If Map(pcpos) <> 2 Then
         If Map(pcpos - 1) <> 4 Then
            pcpos = pcpos - 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
         End If
      End If
   End If
   CheckEnd
End Sub
' Κουμπί για δεξιά
Private Sub cmdRight_Click()
   If pcpos <> 7 And pcpos <> 15 And pcpos <> 23 And pcpos <> 31 And
pcpos <> 39 And pcpos <> 47 And pcpos <> 55 Then
      If Map(pcpos) <> 4 Then
         If Map(pcpos + 1) <> 2 Then
            pcpos = pcpos + 1
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
         End If
      End If
   End If
   CheckEnd
End Sub
' Κουμπί για πάνω
Private Sub cmdUp_Click()
   If pcpos > 7 Then
      If Map(pcpos) <> 3 Then
         If Map(pcpos - 8) <> 5 Then
            pcpos = pcpos - 8
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
         End If
      End If
   End If
   CheckEnd
End Sub
' Μενού - Νέος λαβύρινθος
Private Sub cmdnew_Click()
countL = 0
```

```
    For countL = 0 To 55
        Randomize
        nisse = Int(Rnd * 5 + 1)
        Image1(countL) = LoadPicture(App.Path & "\res\" & nisse & ".bmp")
        Map(countL) = nisse
    Next countL
    ' Αρχικοποίηση θέσεων αρχής και τέλους
    Dim i, j, g
        Randomize Timer
        i = Int(Rnd * 7)
        picend.Top = Image1(i).Top
        picend.Left = Image1(i).Left + Image1(i).Width / 2 - picend.Width / 2
        pcend = i
        Randomize Timer
        j = Int(Rnd * 7)
        PC.Top = Image1(48 + j).Top + Image1(48 + j).Height / 2 - PC.Height / 2
        PC.Left = Image1(48 + j).Left + Image1(48 + j).Width / 2 - PC.Width / 2
        pcpos = 48 + j
        Randomize Timer
        g = Int(Rnd * 7)
        Wumpus.Top = Image1(24 + g).Top + Image1(24 + g).Height / 2 -
Wumpus.Height / 2
        Wumpus.Left = Image1(24 + g).Left + Image1(24 + g).Width / 2 -
Wumpus.Width / 2
        pcwumpus = 24 + g

    ' Αρχικοποίηση χάρτη τεχνητής νοημοσύνης
        For i = 0 To 55
            AImap(i) = 0
        Next i
    ' Αρχικοποίηση χάρτη μέτρησης επαναλήψεων
        For o = 0 To 55
        countmap(o) = 0
        Next o

End Sub
' Υλοποίηση κουμπιών για το παίξιμο
Private Sub Form_KeyUp(KeyCode As Integer, Shift As Integer)
    If KeyCode = vbKeyLeft Then
        cmdLeft_Click
    ElseIf KeyCode = vbKeyRight Then
        cmdRight_Click
    ElseIf KeyCode = vbKeyUp Then
        cmdUp_Click
    ElseIf KeyCode = vbKeyDown Then
        cmdDown_Click
    End If
End Sub
' Εκκίνηση προγράμματος και διαμόρφωση λαβύρινθου
Private Sub Form_Load()
```

```
   cmdnew_Click
      TimerClock = 0
End Sub



Public Sub LearnAImap(a As Byte)
   AImap(a) = 1
   If (a > 7) Then
      AImap(a - 8) = Map(a - 8)
   End If
   If (a < 48) Then
      AImap(a + 8) = Map(a + 8)
   End If
   If a <> 0 And a <> 8 And a <> 16 And a <> 24 And a <> 32 And a <> 40
And a <> 48 Then
      AImap(a - 1) = Map(a - 1)
   End If
   If a <> 7 And a <> 15 And a <> 23 And a <> 31 And a <> 39 And a <> 47
And a <> 55 Then
      AImap(a + 1) = Map(a + 1)
   End If
End Sub

' Έλεγχος για τέλος
Private Sub CheckEnd()
   LearnAImap (pcpos)
   totalsteps = steps + stepsw
   If pcpos = pcend Then

   FormEnd.Show
   FormEnd.Text1 = steps
   FormEnd.Text2 = stepsw
   FormEnd.Text3 = totalsteps

   Pause (800)


   End If
End Sub



Private Sub mnuStart_Click()
   LearnAImap (pcpos)

End Sub
```

```
'Έλεγχος για το αν το τέρας βρήκε τον player
Private Sub CheckKill()
LearnAImap (pcpos)
If pcpos = pcwumpus Then


    GAMEOVER.Show
    GAMEOVER.Text1 = steps
    GAMEOVER.Text2 = stepsw
    GAMEOVER.Text3 = totalsteps
    Pause (800)



End If
End Sub
'Μενού- Μηδενισμός
Private Sub restartzero_Click()
'Μηδενισμός μεταβλητών
steps = 0
stepsw = 0
totalsteps = 0
'Εξαφάνιση των ιχνών του παίκτη
    Image4.Visible = False
    Image5.Visible = False
    Image6.Visible = False
    Image7.Visible = False
    Image2.Visible = False
    Image8.Visible = False
    Image9.Visible = False
    Image10.Visible = False
    Image11.Visible = False
    Image12.Visible = False
    Image13.Visible = False
    Image14.Visible = False
    Image15.Visible = False
    Image16.Visible = False
    Image17.Visible = False
    Image3.Visible = False
    Image18.Visible = False
    Image19.Visible = False
    Image20.Visible = False
    Image21.Visible = False
    Image22.Visible = False
    Image23.Visible = False
    Image24.Visible = False
    Image53.Visible = False
    Image25.Visible = False
    Image26.Visible = False
    Image27.Visible = False
```

```
    Image28.Visible = False
    Image29.Visible = False
    Image30.Visible = False
    Image31.Visible = False
    Image54.Visible = False
    Image32.Visible = False
    Image33.Visible = False
    Image34.Visible = False
    Image35.Visible = False
    Image36.Visible = False
    Image37.Visible = False
    Image38.Visible = False
    Image55.Visible = False
    Image39.Visible = False
    Image40.Visible = False
    Image41.Visible = False
    Image42.Visible = False
    Image43.Visible = False
    Image44.Visible = False
    Image45.Visible = False
    Image56.Visible = False
    Image46.Visible = False
    Image47.Visible = False
    Image48.Visible = False
    Image49.Visible = False
    Image50.Visible = False
    Image51.Visible = False
    Image52.Visible = False
    Image57.Visible = False

'Μηδενισμός χάρτη-μετρητή κινήσεων
countermap.Text1 = 0
countermap.Text8 = 0
countermap.Text15 = 0
countermap.Text22 = 0
countermap.Text29 = 0
countermap.Text36 = 0
countermap.Text43 = 0
countermap.Text50 = 0
countermap.Text2 = 0
countermap.Text9 = 0
countermap.Text16 = 0
countermap.Text23 = 0
countermap.Text30 = 0
countermap.Text37 = 0
countermap.Text44 = 0
countermap.Text51 = 0
countermap.Text3 = 0
countermap.Text10 = 0
countermap.Text17 = 0
```

```
countermap.Text24 = 0
countermap.Text31 = 0
countermap.Text38 = 0
countermap.Text45 = 0
countermap.Text52 = 0
countermap.Text4 = 0
countermap.Text11 = 0
countermap.Text18 = 0
countermap.Text25 = 0
countermap.Text32 = 0
countermap.Text39 = 0
countermap.Text46 = 0
countermap.Text53 = 0
countermap.Text5 = 0
countermap.Text12 = 0
countermap.Text19 = 0
countermap.Text26 = 0
countermap.Text33 = 0
countermap.Text40 = 0
countermap.Text47 = 0
countermap.Text54 = 0
countermap.Text6 = 0
countermap.Text13 = 0
countermap.Text20 = 0
countermap.Text27 = 0
countermap.Text34 = 0
countermap.Text41 = 0
countermap.Text48 = 0
countermap.Text55 = 0
countermap.Text7 = 0
countermap.Text14 = 0
countermap.Text21 = 0
countermap.Text28 = 0
countermap.Text35 = 0
countermap.Text42 = 0
countermap.Text49 = 0
countermap.Text56 = 0

'μηδενισμός χάρτη τεχνητής νοημοσύνης
AI_IconMap.Text1 = 0
AI_IconMap.Text8 = 0
AI_IconMap.Text15 = 0
AI_IconMap.Text22 = 0
AI_IconMap.Text29 = 0
AI_IconMap.Text36 = 0
AI_IconMap.Text43 = 0
AI_IconMap.Text50 = 0
AI_IconMap.Text2 = 0
AI_IconMap.Text9 = 0
AI_IconMap.Text16 = 0
```

```
AI_IconMap.Text23 = 0
AI_IconMap.Text30 = 0
AI_IconMap.Text37 = 0
AI_IconMap.Text44 = 0
AI_IconMap.Text51 = 0
AI_IconMap.Text3 = 0
AI_IconMap.Text10 = 0
AI_IconMap.Text17 = 0
AI_IconMap.Text24 = 0
AI_IconMap.Text31 = 0
AI_IconMap.Text38 = 0
AI_IconMap.Text45 = 0
AI_IconMap.Text52 = 0
AI_IconMap.Text4 = 0
AI_IconMap.Text11 = 0
AI_IconMap.Text18 = 0
AI_IconMap.Text25 = 0
AI_IconMap.Text32 = 0
AI_IconMap.Text39 = 0
AI_IconMap.Text46 = 0
AI_IconMap.Text53 = 0
AI_IconMap.Text5 = 0
AI_IconMap.Text12 = 0
AI_IconMap.Text19 = 0
AI_IconMap.Text26 = 0
AI_IconMap.Text33 = 0
AI_IconMap.Text40 = 0
AI_IconMap.Text47 = 0
AI_IconMap.Text54 = 0
AI_IconMap.Text6 = 0
AI_IconMap.Text13 = 0
AI_IconMap.Text20 = 0
AI_IconMap.Text27 = 0
AI_IconMap.Text34 = 0
AI_IconMap.Text40 = 0
AI_IconMap.Text47 = 0
AI_IconMap.Text54 = 0
AI_IconMap.Text7 = 0
AI_IconMap.Text14 = 0
AI_IconMap.Text21 = 0
AI_IconMap.Text28 = 0
AI_IconMap.Text35 = 0
AI_IconMap.Text42 = 0
AI_IconMap.Text49 = 0
AI_IconMap.Text56 = 0

countL = 0
   For countL = 0 To 55
      Randomize
      nisse = Int(Rnd * 5 + 1)
```

```
        Image1(countL) = LoadPicture(App.Path & "\res\" & nisse & ".bmp")
        Map(countL) = nisse
    Next countL
    ' Αρχικοποίηση θέσεων αρχής και τέλους
    Dim i, j, g
        Randomize Timer
        i = Int(Rnd * 7)
        picend.Top = Image1(i).Top
        picend.Left = Image1(i).Left + Image1(i).Width / 2 - picend.Width / 2
        pcend = i
        Randomize Timer
        j = Int(Rnd * 7)
        PC.Top = Image1(48 + j).Top + Image1(48 + j).Height / 2 - PC.Height / 2
        PC.Left = Image1(48 + j).Left + Image1(48 + j).Width / 2 - PC.Width / 2
        pcpos = 48 + j
        Randomize Timer
        g = Int(Rnd * 7)
        Wumpus.Top = Image1(24 + g).Top + Image1(24 + g).Height / 2 -
Wumpus.Height / 2
        Wumpus.Left = Image1(24 + g).Left + Image1(24 + g).Width / 2 -
Wumpus.Width / 2
        pcwumpus = 24 + g

    ' Αρχικοποίηση χάρτη τεχνητής νοημοσύνης
        For i = 0 To 55
            AImap(i) = 0
        Next i
    ' Αρχικοποίηση χάρτη μέτρησης επαναλήψεων
        For o = 0 To 55
        countmap(o) = 0
        Next o

End Sub


Private Sub tmrpause_Timer()
    TimerClock = TimerClock + 1
End Sub
'Καθυστέρηση της συχνότητας κίνησης
Public Sub Pause(x As Integer)
    Dim nowtime As Double
    nowtime = TimerClock
    Do
        DoEvents
   Loop Until TimerClock > nowtime + x
End Sub

Public Sub trace()

    If pcpos = 1 Then
```

```
    Image4.Visible = True
    End If

    If pcpos = 2 Then
    Image5.Visible = True
End If
If pcpos = 3 Then
    Image6.Visible = True
End If
If pcpos = 4 Then
    Image7.Visible = True

End If

If pcpos = 0 Then
    Image2.Visible = True

End If

If pcpos = 5 Then
    Image8.Visible = True

End If

If pcpos = 6 Then
    Image9.Visible = True

End If

If pcpos = 7 Then
    Image10.Visible = True

End If

If pcpos = 8 Then
    Image11.Visible = True

End If

If pcpos = 9 Then
    Image12.Visible = True

End If

If pcpos = 10 Then
    Image13.Visible = True

End If

If pcpos = 11 Then
```

```
    Image14.Visible = True

End If

If pcpos = 12 Then
    Image15.Visible = True

End If

If pcpos = 13 Then
    Image16.Visible = True

End If

If pcpos = 14 Then
    Image17.Visible = True

End If

If pcpos = 15 Then
    Image3.Visible = True

End If

If pcpos = 16 Then
    Image18.Visible = True

End If
If pcpos = 17 Then
    Image19.Visible = True

End If

If pcpos = 18 Then
    Image20.Visible = True

End If

If pcpos = 19 Then
    Image21.Visible = True

End If

If pcpos = 20 Then
    Image22.Visible = True

End If

If pcpos = 21 Then
    Image23.Visible = True
```

```
End If

If pcpos = 22 Then
   Image24.Visible = True

End If

If pcpos = 23 Then
   Image53.Visible = True

End If

If pcpos = 24 Then
   Image25.Visible = True

End If

If pcpos = 25 Then
   Image26.Visible = True

End If

If pcpos = 26 Then
   Image27.Visible = True

End If


If pcpos = 27 Then
   Image28.Visible = True

End If

If pcpos = 28 Then
   Image29.Visible = True

End If

If pcpos = 29 Then
   Image30.Visible = True

End If

If pcpos = 30 Then
   Image31.Visible = True

End If

If pcpos = 31 Then
```

```
    Image54.Visible = True

End If

If pcpos = 32 Then
    Image32.Visible = True

End If

If pcpos = 33 Then
    Image33.Visible = True

End If

If pcpos = 34 Then
    Image34.Visible = True

End If

If pcpos = 35 Then
    Image35.Visible = True

End If
If pcpos = 36 Then
    Image36.Visible = True

End If

If pcpos = 37 Then
    Image37.Visible = True

End If

If pcpos = 38 Then
    Image38.Visible = True

End If

If pcpos = 39 Then
    Image55.Visible = True

End If

If pcpos = 40 Then
    Image39.Visible = True

End If

If pcpos = 41 Then
    Image40.Visible = True
```

```
End If

If pcpos = 42 Then
   Image41.Visible = True

End If

If pcpos = 43 Then
   Image42.Visible = True

End If

If pcpos = 44 Then
   Image43.Visible = True

End If

If pcpos = 45 Then
   Image44.Visible = True

End If

If pcpos = 46 Then
   Image45.Visible = True

End If

If pcpos = 47 Then
   Image56.Visible = True

End If

If pcpos = 48 Then
   Image46.Visible = True

End If

If pcpos = 49 Then
   Image47.Visible = True

End If

If pcpos = 50 Then
   Image48.Visible = True

End If

If pcpos = 51 Then
   Image49.Visible = True
```

```
End If

If pcpos = 52 Then
   Image50.Visible = True

End If

If pcpos = 53 Then
   Image51.Visible = True

End If

If pcpos = 54 Then
   Image52.Visible = True

End If

If pcpos = 55 Then
   Image57.Visible = True

End If


End Sub

Public Sub counttimes()

countmap(pcpos) = countmap(pcpos) + 1

End Sub

Public Sub checkup()
If pcpos > 7 Then
If countmap(pcpos - 8) >= 5 Then
vroc = 11
End If
End If
End Sub

Public Sub checkright()
If pcpos < 55 Then
If countmap(pcpos + 1) >= 5 Then
vroc = 11
End If
End If
End Sub

Public Sub checkleft()
If pcpos > 1 Then
```

```
If countmap(pcpos - 1) >= 5 Then
vroc = 11
End If
End If
End Sub

Public Sub checkdown()
If pcpos <= 47 Then
If countmap(pcpos + 8) >= 5 Then
vroc = 11
End If
End If
End Sub

Public Sub showcounter()
If countermap.Visible = True Then

countermap.Text1 = countmap(0)
countermap.Text8 = countmap(1)
countermap.Text15 = countmap(2)
countermap.Text22 = countmap(3)
countermap.Text29 = countmap(4)
countermap.Text36 = countmap(5)
countermap.Text43 = countmap(6)
countermap.Text50 = countmap(7)
countermap.Text2 = countmap(8)
countermap.Text9 = countmap(9)
countermap.Text16 = countmap(10)
countermap.Text23 = countmap(11)
countermap.Text30 = countmap(12)
countermap.Text37 = countmap(13)
countermap.Text44 = countmap(14)
countermap.Text51 = countmap(15)
countermap.Text3 = countmap(16)
countermap.Text10 = countmap(17)
countermap.Text17 = countmap(18)
countermap.Text24 = countmap(19)
countermap.Text31 = countmap(20)
countermap.Text38 = countmap(21)
countermap.Text45 = countmap(22)
countermap.Text52 = countmap(23)
countermap.Text4 = countmap(24)
countermap.Text11 = countmap(25)
countermap.Text18 = countmap(26)
countermap.Text25 = countmap(27)
countermap.Text32 = countmap(28)
countermap.Text39 = countmap(29)
countermap.Text46 = countmap(30)
countermap.Text53 = countmap(31)
countermap.Text5 = countmap(32)
```

```
countermap.Text12 = countmap(33)
countermap.Text19 = countmap(34)
countermap.Text26 = countmap(35)
countermap.Text33 = countmap(36)
countermap.Text40 = countmap(37)
countermap.Text47 = countmap(38)
countermap.Text54 = countmap(39)
countermap.Text6 = countmap(40)
countermap.Text13 = countmap(41)
countermap.Text20 = countmap(42)
countermap.Text27 = countmap(43)
countermap.Text34 = countmap(44)
countermap.Text41 = countmap(45)
countermap.Text48 = countmap(46)
countermap.Text55 = countmap(47)
countermap.Text7 = countmap(48)
countermap.Text14 = countmap(49)
countermap.Text21 = countmap(50)
countermap.Text28 = countmap(51)
countermap.Text35 = countmap(52)
countermap.Text42 = countmap(53)
countermap.Text49 = countmap(54)
countermap.Text56 = countmap(55)

End If

End Sub


Public Sub pclookup()

If pcpos > 7 Then
      If Map(pcpos) <> 3 Then
         If Map(pcpos - 8) <> 5 Then
            If pcpos - 8 = pcwumpus Then
            pcpos = pcpos - 8
            PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
            PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
            End If
         End If
      End If
   End If
   CheckEnd
End Sub


Public Sub lookleft()
```

```
If pcpos <> 0 And pcpos <> 8 And pcpos <> 16 And pcpos <> 24 And pcpos
<> 32 And pcpos <> 40 And pcpos <> 48 Then
       If Map(pcpos) <> 2 Then
          If Map(pcpos - 1) <> 4 Then
           If pcpos - 1 <> pcwumpus Then
             If pcpos - 1 = pcend Then
             pcpos = pcpos - 1
             PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
             PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
          End If
       End If
       End If
       End If
   End If
   CheckEnd
End Sub

Public Sub lookright()

If pcpos <> 7 And pcpos <> 15 And pcpos <> 23 And pcpos <> 31 And pcpos
<> 39 And pcpos <> 47 And pcpos <> 55 Then
       If Map(pcpos) <> 4 Then
          If Map(pcpos + 1) <> 2 Then
          If pcpos + 1 <> pcwumpus Then
          If pcpos + 1 = pcend Then

             pcpos = pcpos + 1
             PC.Top = Image1(pcpos).Top + Image1(pcpos).Height / 2 -
PC.Height / 2
             PC.Left = Image1(pcpos).Left + Image1(pcpos).Width / 2 -
PC.Width / 2
          End If
       End If
       End If
       End If

   End If
   trace
   counttimes
   showcounter
   CheckEnd
End Sub

Public Sub lookforplayer()
'wumpus επιτίθεται δεξιά
If (pcwumpus <> 7 And pcwumpus <> 15 And pcwumpus <> 23 And
pcwumpus <> 31 And pcwumpus <> 39 And pcwumpus <> 47 And pcwumpus
<> 55) Then
```

```
   If (Map(pcwumpus) <> 4 And Map(pcwumpus + 1) <> 2) Then

      If pcwumpus > 7 Then

            If pcwumpus + 1 = pcpos Then
             pcwumpus = pcwumpus + 1
             stepsw = stepsw + 1
            Wumpus.Top = Image1(pcwumpus).Top +
Image1(pcwumpus).Height / 2 - Wumpus.Height / 2
            Wumpus.Left = Image1(pcwumpus).Left +
Image1(pcwumpus).Width / 2 - Wumpus.Width / 2

Pause (0.15)
         showcounter


            CheckEnd
CheckKill
End If
End If
End If

End If

'wumpus επιτίθεται πάνω

If pcwumpus > 7 Then
   If Map(pcwumpus) <> 3 Then
      If (Map(pcwumpus - 8) <> 5) Then
      If pcwumpus - 8 = pcpos Then
pcwumpus = pcwumpus - 8
stepsw = stepsw + 1
            Wumpus.Top = Image1(pcwumpus).Top +
Image1(pcwumpus).Height / 2 - Wumpus.Height / 2
            Wumpus.Left = Image1(pcwumpus).Left +
Image1(pcwumpus).Width / 2 - Wumpus.Width / 2
Pause (0.15)
            showcounter
            CheckEnd
CheckKill
trace
End If
End If
End If
End If


'wumpus επιτίθεται αριστερά
If (pcwumpus <> 0 And pcwumpus <> 8 And pcwumpus <> 16 And
pcwumpus <> 24 And pcwumpus <> 32 And pcwumpus <> 40 And pcwumpus
```

```
<> 48) Or (pcwumpus = 7 Or pcwumpus = 15 Or pcwumpus = 23 Or
pcwumpus = 31 Or pcwumpus = 39 Or pcwumpus = 47 Or pcwumpus = 55)
Then
  If Map(pcwumpus) <> 2 Then
  If Map(pcwumpus - 1) <> 4 Then
  If pcwumpus - 1 = pcpos Then
  pcwumpus = pcwumpus - 1
  stepsw = stepsw + 1
            Wumpus.Top = Image1(pcwumpus).Top +
Image1(pcwumpus).Height / 2 - Wumpus.Height / 2
            Wumpus.Left = Image1(pcwumpus).Left +
Image1(pcwumpus).Width / 2 - Wumpus.Width / 2
          Pause (0.15)

            showcounter
            CheckEnd
CheckKill
trace
End If
End If
End If
End If

'wumpus επιτίθεται κάτω

If pcwumpus < 48 Then
If (Map(pcwumpus) <> 5 And Map(pcwumpus + 8) <> 3) Then
If pcwumpus + 8 = pcpos Then
pcwumpus = pcwumpus + 8
stepsw = stepsw + 1
            Wumpus.Top = Image1(pcwumpus).Top +
Image1(pcwumpus).Height / 2 - Wumpus.Height / 2
            Wumpus.Left = Image1(pcwumpus).Left +
Image1(pcwumpus).Width / 2 - Wumpus.Width / 2
Pause (0.15)
lookforplayer
pclookup
        showcounter


            CheckEnd
CheckKill
trace
End If
End If
End If


hell:
End Sub
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

2) Science Illustrated Τεύχος 11, Φεβρουάριος 2006, σελ. 26-31
3) E. Petroutsos ,Visual Basic 6,Εκδόσεις Γκιούρδας Μ.
4) Stuart Russell & Peter Norvig, 2003,Artificial Intelligence-A Modern Approach,Second Edition, Prentice Hall
5) http://www.ccg.leeds.ac.uk/james/aStar
6) http://www.gameai.com
7) PHA*: Finding the Shortest Path with A* in An Unknown Physical Environment, Journal of Artificial Intelligence Research 21 (2004) 631-670 Submitted 9/03; published 6/04
8) Ballard,B.W(1983). The *-minimax search procedure for trees containing chance nodes. Artificial Intelligence,σελ. 327-350
9) Deng, X. and Papadimitriou,C.H. (1990). Exploring an unknown graph. In proceedings 31st Annual Symposium on Foundations of Computer Science, pp 355-361, St Louis. IEEE Computer Society Press
10) Beal D. F. (1980), An analysis on Minimax. In Clarke M.R.B., Advances in Computer chess 2, pp 103-109, Edinburgh University Press, Edinburgh, Scotland
11) Warren,D.H.D.(1974).WAPLAN: A system for generating plans. Department of Computational Logic Memo 76,University of Edinburgh, Scotland
12) McCulloch,W.S. and Pitts, W.(1943), A logical calculus of the ideas immanent in nervous activity, Bulletin of Mathematical Biophysics, 5, 115-137
13) Whitehead, A.N. (1911).An introduction to Mathematics. Williams and Northgate, London
14) Hebb, D. O. (1949). The organization of behavior. Wiley, New York
15) McCarthy, J., Minsky, M.L., Rochester, N., and Shannon, C., E., (1955). Proposal for the Dartmouth summer research project on artificial intelligence, Tech. Rep., Dartmouth College
16) Allen, J., F., (1984) Towards a general theory of action and time, Artificial Intelligence, 23, 123-154

17) Newell, A., Shaw, J, C, and Simon , H. A (1958). Chess playing programs and the problem of complexity. IBM journal of research and development, 4(2), 320-335
18) Gelertner, H. (1959). Realization of a geometry theorem proving machine. In proceedings of an international conference on information proceeding. pp 273-282, Paris, UNESCO House
19) Samuel, A., L., (1959).Some studies in machine learning using the game of checkers. IBM journal of research and development, 3(3), 210-229.
20) Minsky, M.L. (1968). Semantic information processing. MIT Press, Cambridge, Massachussetts
21) Robinson, J. A. (1965). A machine oriented logic based on the resolution principle. Journal of the association for information science,12, 23-41
22) Slagle, J. R., (1963a). A heuristic program that solves symbolic integration problems in freshman calculus. Journal of the association for computing machinery, 10 (4)
23) Evans T.G., (1968) A program for the solution of a class of a geometric-analogy intelligence-test questions. In Minsky , M. L. Semantic information processing. MIT Press, Cambridge, Massachussetts
24) Bobrow, D. G., (1967). Natural language input for a computer problem solving system. In Minsky , M. L. Semantic information processing. 133-215 MIT Press, Cambridge, Massachussetts
25) Buchanan,B.G., Sutherland, G.L. and Feigenbaum, E.A,(1969). Heuristic DENDRAL: a program for generating explanatory hypotheses in organic chemistry. In Meltzer, B., Michie and Swann, machine intelligence 4, 209-254 Edinburgh University Press, Edinburgh, Scotland
26) Hopfield, J. J. (1982). Neurons with graded response have collective computational properties like those of two-state neurons. Proceedings of the national academy of sciences of the united states of America. 79, 2554-2558
27) Rumelhart, D.E., Hinton, G.E., and Williams, R.J., (1986b). Learning represantations by propagating errors. Nature, 323, 533-536
28) Bryson and Ho,(1969), Applied optimal control, Blaisdell, New York
29) Cohen, W.W., and Page, C,D. (1995). Learnability in inductive logic programming : methods and results. New generation computing

30) Laird, J. E., Rosenbloom, Newell(1986), Chunking in Soar: The anatomy of a general learning mechanism, Machine Learning,1,11-46
31) Korf R. E. Linear space best first search. Artificial Intelligence 42(3),189-212