Πολυτεχνειο Κρητης
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

ΠΛΗ 311 – Τεχνητή Νοημοσύνη – 2024

Χειμερινό Εξάμηνο 2024 **Εργασία Προγραμματισμού** Παράδοση: **08.02.2025**

Παίζοντας το Παιχνίδι HexThello

Επιμέλεια εργασίας: Ιωάννης Σκουλάκης, Μιχαήλ Γ. Λαγουδάκης

1 Εισαγωγή

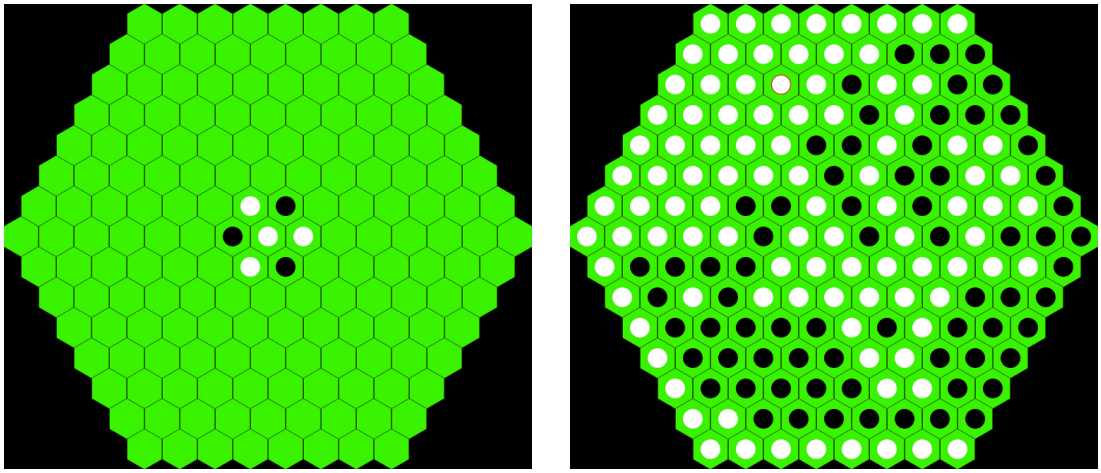
Στο πλαίσιο της εργασίας προγραμματισμού του μαθήματος θα σχεδιάσετε και θα υλοποιήσετε ένα πρόγραμμα που παίζει το παιχνίδι HexThello. Στο παρόν κείμενο υπάρχει ένας οδηγός με τους κανόνες του παιχνιδιού, καθώς και μια σύντομη περιγραφή του κώδικα που δίνεται ως βάση.

Η εργασία είναι ατομική και είστε ελεύθεροι να χρησιμοποιήσετε όποια γλώσσα προγραμματισμού θέλετε, αρκεί να υπάρχει συμβατότητα επικοινωνίας με τον server του παιχνιδιού που σας δίνεται. Μετά την παράδοση των εργασιών και ανάλογα με τον αριθμό των συμμετοχών, ίσως διεξαχθεί ένα πρωτάθλημα HexThello μεταξύ όλων των παικτών. Σ΄ αυτήν την περίπτωση, θα δοθεί επιπλέον βαθμολογία (bonus) από 0% έως 20% του βαθμού της εργασίας ανάλογα με την τελική κατάταξη του παίκτη σας.

2 Το Παιχνίδι

Το HexThello παίζεται με λευκούς και μαύρους δίσκους πάνω σε έναν εξαγωνικό πίνακα (ταμπλώ) με 169 κυψέλες (θέσεις) συνολικά με την κάθε πλευρά στην περιφέρεια να έχει 8 κυψέλες. Το παιχνίδι ξεκινάει με μια αρχική στοιχειοθέτηση 4 λευκών και 3 μαύρων δίσκων στο κέντρο του πίνακα, όπως φαίνεται στο Σχήμα 1 (α). Ο στόχος κάθε παίκτη είναι να έχει στο ταμπλώ περισσότερους δίσκους του δικού του χρώματος στο τέλος του παιχνιδιού, όπως φαίνεται στο Σχήμα 1 (β). Στη συνέχεια περιγράφουμε τους κανόνες του παιχνιδιού.

- Οι παίκτες παίζουν εναλλάξ και ο μαύρος παίκτης παίζει πρώτος.
- Ο λευκός παίκτης τοποθετεί λευκούς δίσκους και ο μαύρος παίκτης μαύρους δίσκους.
- Κάθε κίνηση αποτελείται από την τοποθέτηση ενός δίσκου σε κάποια θέση του πίνακα με τέτοιον τρόπο ώστε να εγκλωβίζει κάποιους δίσκους του αντιπάλου (τουλάχιστον έναν).
- Εγκλωβισμός σημαίνει μια ακολουθία συνεχόμενων δίσκων του αντιπάλου σε ευθεία (σε οποιαδήποτε από τις τρεις διευθύνσεις) να βρεθεί ανάμεσα σε δύο δίσκους του παίκτη.
- Οι εγκλωβισμένοι κόμβοι αλλάζουν χρώμα και περνούν στον παίκτη που μόλις έπαιξε.
- Μια κίνηση (τοποθέτηση δίσκου) μπορεί να εγκλωβίσει οποιονδήποτε αριθμό αντίπαλων δίσκων την ίδια στιγμή σε οποιαδήποτε από τις έξι κατευθύνσεις από την θέση της κίνησης.
- Οι δίσκοι μπορούν να εγκλωβισθούν μόνο σαν άμεσο αποτέλεσμα μιας κίνησης.
- Όλοι οι δίσκοι που εγκλωβίζονται πρέπει να αλλάζουν χρώμα υποχρεωτικά.
- Αν ένας παίκτης δεν μπορεί να εγκλωβίσει δίσκους του αντιπάλου, χάνει τη σειρά σου.
- Όταν δεν υπάρχει καμία διαθέσιμη κίνηση και για τους δύο παίχτες, το παιχνίδι τελειώνει.
- Στο τέλος του παιχνιδιού, κερδίζει ο παίκτης με τους περισσότερους δίσκους στο ταμπλώ.



Σχήμα 1: (α) Αρχική κατάσταση, (β) Τελική κατάσταση με νίκη των λευκών (99 έναντι 70).

3 Διαδικαστικά

Η διαδικασία που πρέπει να ακολουθήσετε για να ολοκληρώσετε την εργασία είναι η παρακάτω:

- Διαβάστε προσεκτικά το παρόν κείμενο και τον κώδικα που το συνοδεύει.
- Βεβαιωθείτε ότι μπορείτε να τρέξετε τον κώδικα και κατανοήστε τη λειτουργία του.
- Υλοποιήστε τον αλγόριθμο αναζήτησης minimax .
- Υλοποιήστε μια καλή συνάρτηση αξιολόγησης και μια μέθοδο αποκοπής.
- Βελτιώστε την αναζήτηση υλοποιώντας α–β pruning.
- Βελτιώστε περαιτέρω την αναζήτηση με singular extensions, forward pruning, κλπ.
- Βεβαιωθείτε ότι το πρόγραμμά σας συνεργάζεται με τον server χωρίς προβλήματα.
- Παραδώστε τον κώδικα με την εργασία σας (zip) μέχρι την ημερομηνία παράδοσης.
- Παραδώστε μια σύντομη αναφορά (pdf) για τις τεχνικές που υλοποιήσατε.
- Παρουσιάστε την εργασία σας στον διδάσκοντα του μαθήματος.
- Λάβετε μέρος στο πρωτάθλημα και κερδίστε με τον παίκτη σας!

4 Εκπόνηση

Ο αλγόριθμος αναζήτησης minimax και το α–β pruning, καθώς και άλλες τεχνικές για παιχνίδια έχουν καλυφθεί εκτενώς στο μάθημα. Επίσης, τα βιβλία του μαθήματος περιέχουν όλες τις λεπτομέρειες και τον ψευδοκώδικα των αλγορίθμων αναζήτησης που χρειάζεστε για την εργασία.

Σκεφτείτε τον τρόπο με τον οποίο θα δημιουργήσετε το δέντρο της αναζήτησης. Η μέθοδος που θα ακολουθήσετε μπορεί να επηρεάσει σημαντικά την ταχύτητα του παίκτη σας. Προφανώς, δεν θα υπάρχει αρκετός χρόνος για να να ψάξει το πρόγραμμά σας ολόκληρο το δέντρο αναζήτησης για την καλύτερη κίνηση. Έτσι, θα πρέπει να ψάχνετε για την καλύτερη δυνατή κίνηση σε λογικά πλαίσια πραγματικού χρόνου. Ακολουθούν κάποια σχόλια και προτάσεις.

- Βεβαιωθείτε ότι μετά την υλοποίηση του α–β pruning η αναζήτησή σας καταλήγει ακριβώς στις ίδιες τιμές με τον minimax. Θα ήταν δόκιμο να έχετε ένα μηχανισμό για να ενεργοποιείτε και να απενεργοποιείτε το α–β pruning κατά βούληση.

- Ο απλός αλγόριθμος α–β pruning εξετάζει τα παιδιά κάθε κόμβου με τη σειρά δημιουργίας, αλλά μπορείτε να υλοποιήσετε κάτι πιο έξυπνο. Όπως ξέρετε, η σειρά με την οποία γίνονται επεκτάσεις κόμβων έχει μεγάλη επιρροή στην αποτελεσματικότητα του κλαδέματος.
- Μια αρχική (απλή) συνάρτηση αξιολόγησης κατάστασης (state) που μπορείτε να χρησιμοποιήσετε σε συνδυασμό με κάποια μέθοδο αποκοπής είναι η παρακάτω:

$$V(\text{state}) = (\text{number of my pieces}) - (\text{number of opponent's pieces})$$

- Μπορείτε να χρησιμοποιήσετε οποιαδήποτε τεχνική βελτίωσης βρείτε διαθέσιμη στο διαδίκτυο ή σε βιβλία αρκεί να εξηγήσετε στην αναφορά σας τι κάνει, πως το κάνει, γιατί την επιλέξατε και πως την προσαρμόσατε στον κώδικά σας.
- Στην αναφορά σας περιγράψτε τις ιδιαιτερότητες του κώδικά σας και τις τυχόν προεκτάσεις ή έξυπνες λύσεις που υλοποιήθηκαν. Θα πρέπει απαραίτητα να υπάρχουν αναφορές των πηγών σας· τυφλή αντιγραφή (plagiarism) θα επιφέρει συνέπειες.

5 Υλοποίηση

Μπορείτε να γράψετε τον κώδικα σας ως συνέχεια του κώδικα TUC HexThello που σας δίνεται (ενότητα **Έγγραφα→Εργασίες 2024** στο eClass). Εκεί υπάρχει υλοποιημένη η απαραίτητη λειτουργικότητα του παιχνιδιού, ώστε να ελαττωθεί ο φόρτος υλοποίησης. Συγκεκριμένα, περιλαμβάνεται ο κώδικας ενός βασικού παίκτη (client) και του server του παιχνιδιού σε C. Για να δημιουργήσετε και να τρέξετε τα εκτελέσιμα αρχεία, θα πρέπει να κάνετε `make` στον υπολογιστή σας (δείτε σχετικές οδηγίες στο αρχείο `README`).

Ο κώδικας παρέχει τις βασικές δομές δεδομένων και τους βασικούς μηχανισμούς για το παιχνίδι. Επίσης, υπάρχουν υλοποιημένες και βοηθητικές συναρτήσεις (π.χ. έλεγχος αν μια κίνηση είναι έγκυρη), χωρίς αυτό όμως να σημαίνει ότι δεν μπορεί κάποιος να υλοποιήσει τις δικές του συναρτήσεις. **Προσοχή!** Δεν επιτρέπονται αλλαγές στο πρωτόκολλο επικοινωνίας. Αν αλλάξετε τον τρόπο επικοινωνίας, η συμμετοχή σας στο τουρνουά δεν θα είναι εφικτή.

5.1 Υποδομή

Οι δομές που χρησιμοποιούνται στην υπάρχουσα υλοποίηση βρίσκονται στα αρχεία `board.h` και `move.h`. Στο πρώτο αρχείο ορίζεται η δομή `Position` που αναπαριστά μια πλήρη κατάσταση του παιχνιδιού. Η δομή αυτή εμπεριέχει έναν διδιάστατο πίνακα `board` μεγέθους `ARRAY_BOARD_SIZE`×`ARRAY_BOARD_SIZE` που αναπαριστά την διάταξη των πιονιών στο εξάγωνο ταμπλώ (`0` για λευκό πιόνι, `1` για μαύρο πιόνι, `2` για κενό τετράγωνο, `4` για αχρησιμοποίητο τετράγωνο), έναν μονοδιάστατο πίνακα `score` με την τρέχουσα βαθμολογία (μία τιμή για κάθε παίκτη) και μία μεταβλητή `turn` που δηλώνει την ταυτότητα του παίκτη (`0` για τον λευκό, `1` για τον μαύρο) που καλείται να παίξει στην εν λόγω διάταξη. Στο δεύτερο αρχείο ορίζεται η δομή `Move` που χρησιμοποιείται για να αναπαριστά μια κίνηση του παιχνιδιού. Σ' αυτήν περιέχονται μέσα σε έναν πίνακα `tile` οι συντεταγμένες της θέσης που επιλέξαμε για την κίνησή μας, καθώς και η ταυτότητα `color` του παίκτη στον οποίο ανήκει η κίνηση.

Όπως αναφέραμε παραπάνω, η αποθήκευση ολόκληρου του εξάγωνου ταμπλώ στην μνήμη γίνεται με την βοήθεια ενός διδιάστατου πίνακα. Η λογική που ακολουθείται συναντάτε σε πολλά παιχνίδια με τέτοιου είδους ταμπλώ και περιγράφεται λεπτομερώς στον σύνδεσμο http://www.redblobgames.com/grids/hexagons/#map-storage. Αναλυτικότερα, σώζουμε κάθε γραμμή του εξάγωνου ταμπλώ μας σε μία γραμμή μέσα στον διδιάστατο πίνακα, με μία ιδιαιτερότητα. Ο πίνακάς μας έχει μέγεθος `ARRAY_BOARD_SIZE`×`ARRAY_BOARD_SIZE`, συνεπώς το εξάγωνο ταμπλώ μας που έχει μεγαλύτερη (μεσαία) γραμμή ίση με `ARRAY_BOARD_SIZE` θα αφήνει αναγκαστικά κάποιες αχρησιμοποίητες θέσεις. Αυτές οι θέσεις για τις πρώτες μισές γραμμές βρίσκονται στην αρχή τους, ενώ για τις δεύτερες μισές στο τέλος τους. Αυτό σημαίνει ότι οι πρώτες (`ARRAY_BOARD_SIZE`$-1)/2$ θέσεις της πρώτης γραμμής, οι πρώτες (`ARRAY_BOARD_SIZE`$-1)/2)-1$ θέσεις της δεύτερης, κ.ο.κ. αγνοούνται. Αντίστοιχα, όταν περάσουμε το μέσο του ταμπλώ μας,

αυτές οι θέσεις βρίσκονται στο τέλος της εκάστοτε γραμμής. Ένας σημαντικός λόγος για τον οποίο υιοθετείται αυτή η αναπαράσταση είναι το γεγονός ότι μπορούμε εύκολα, παίρνοντας την οριζόντια, κάθετη και διαγώνια διεύθυνση στον δισδιάστατο πίνακα (χωρίς τις θέσεις που αγνοούνται), να περιπλανηθούμε στις «νόμιμες» διευθύνσεις του εξάγωνου ταμπλώ.

Όσον αφορά στην δομή με την οποία αποθηκεύουμε μία κίνηση, θα πρέπει να διευκρινιστεί ότι σε περίπτωση που δεν έχουμε καμία κίνηση διαθέσιμη, τότε πρέπει να στείλουμε την κενή (null) κίνηση, η οποία στην ουσία είναι μία κίνηση με `tile[0]` ίσο με `NULL_MOVE`.

Είστε ελεύθεροι να κάνετε όποιες αλλαγές θέλετε σ' αυτές τις δομές, ακόμα και να τις αντικαταστήσετε με δικές σας. Πρέπει όμως να προσέχετε ώστε οι αλλαγές σας να μην επηρεάσουν την επικοινωνία του παίκτη σας με τον server του παιχνιδιού.

5.2 Χρήσιμες Διεργασίες

Στο αρχείο `board.h` υπάρχουν δηλωμένες μερικές χρήσιμες διεργασίες και συναρτήσεις ελεύθερες προς χρήση τις οποίες και περιγράφουμε συνοπτικά παρακάτω.

- `void doMove( Position * pos, Move * moveToDo )`
  Η διεργασία αυτή, εκτελεί την κίνηση που ορίζεται μέσα στο `moveToDo` πάνω στην κατάσταση που ορίζεται στο `pos`.

- `int canMove( Position * pos, char color )`
  Αυτή η συνάρτηση, λαμβάνοντας υπόψη το ταμπλώ και τον παίκτη player της κατάστασης `pos` επιστρέφει `1`, αν ο παίκτης αυτός έχει τουλάχιστον μία νόμιμη κανονική κίνηση στο εν λόγω ταμπλώ, ή `0`, σε περίπτωση που δεν έχει καμία (συνεπώς θα πρέπει υποχρεωτικά να αποστείλει την `NULL_MOVE`).

- `int isLegalMove( Position * pos, Move * moveToCheck )`
  Η συνάρτηση αυτή επιστρέφει `1`, αν η κίνηση `moveToCheck` είναι νόμιμη στο ταμπλώ της κατάστασης `pos`, διαφορετικά επιστρέφει `0`. Προσοχή, η συνάρτηση αυτή δεν θεωρεί την `NULL_MOVE` ως νόμιμη, αλλά εξετάζει μόνο τις κανονικές κινήσεις. Συνεπώς, ανάλογα με το τι θέλουμε να κάνουμε, μπορεί να πρέπει να χρησιμοποιηθεί μαζί με την συνάρτηση `canMove`.

5.3 Επικοινωνία

Για να μπορούν δύο προγράμματα να παίξουν μεταξύ τους, έχει υλοποιηθεί μια απλή client–server αρχιτεκτονική. Ο server επικοινωνεί με δύο clients που αντιπροσωπεύουν τους δύο παίκτες. Οι clients μπορούν να βρίσκονται στο ίδιο ή ακόμα και σε διαφορετικά μηχανήματα. Ο server αρχικά ενημερώνει τους παίκτες για την κατάσταση του παιχνιδιού (ταμπλώ, βαθμολογία, παίκτης) και για το χρώμα τους. Στη συνέχεια, σε κάθε στιγμή ο server στέλνει στον έναν από τους δύο clients (σ' αυτόν που έχει σειρά να παίξει) την τελευταία κίνηση του αντιπάλου του. Ο client πρέπει να αποφασίσει την κίνηση του **μέσα σε λογικά χρονικά πλαίσια** και να στείλει ένα μήνυμα με την κίνησή του, πίσω στο server. Ο κάθε client οφείλει να ενημερώνει εσωτερικά την κατάσταση του παιχνιδιού βάσει της αρχικής κατάστασης και των κινήσεων που έχουν γίνει από τον ίδιο και τον αντίπαλο. Όλες οι ρουτίνες επικοινωνίας μεταξύ client και server είναι υλοποιημένες. **Δεν επιτρέπονται αλλαγές σ' αυτές τις ρουτίνες!!!** Υπάρχει ένας απλός client στο αρχείο `client.c` που παίζει με τυχαίες κινήσεις. Αυτός ο κώδικας μπορεί να χρησιμοποιηθεί ως βάση για να υλοποιηθεί ο παίκτης σας. **Λογικά, δεν θα χρειαστεί να πειράξετε κανένα άλλο αρχείο, πλην του `client.c`.**

Σε περίπτωση που επιθυμείτε να δουλέψετε σε κάποια άλλη γλώσσα πέραν της C, πρέπει να υλοποιήσετε το ίδιο πρωτόκολλο επικοινωνίας στην πλευρά του δικού σας client, ώστε να είστε συμβατοί με τον server. Σ' αυτήν την περίπτωση θα σας βοηθήσει ο υπάρχων κώδικας και η λεπτομερής παρουσίαση της επικοινωνίας που ακολουθεί.

Μέσα στο αρχείο `comm.h` ορίζονται επτά (7) τύποι μηνυμάτων που αποστέλλονται μεταξύ του server και του client και καθορίζουν την επικοινωνία, καθώς και οι συναρτήσεις που αποστέλλουν και λαμβάνουν δεδομένα. Όλα τα μηνύματα που ανταλλάσσονται, εκτός από αυτό της μεταφοράς

του ονόματος του παίκτη, έχουν σταθερό μήκος. Ο server είναι αυτός που έχει τον απόλυτο έλεγχο της επικοινωνίας και ο ρόλος του client είναι να υπακούει και να αντιδρά σωστά στις όποιες οδηγίες του. Μετά τη δημιουργία σύνδεσης μεταξύ server και client (μετά το accept), ο client πρέπει να περιμένει την αποδοχή κάποιου μηνύματος (από τα επτά που αναφέρθηκαν παραπάνω). Η αντίδρασή του ανάλογα με το μήνυμα του server πρέπει να είναι η ακόλουθη:

- `NM_NEW_POSITION`
  Ο server μας ενημερώνει ότι θα μας αποστείλει νέα δομή `Position`. Η αντίδρασή μας πρέπει να είναι η εκτέλεση της διεργασίας `void getPosition( Position * posToGet, int mySocket )`. Στην ουσία έτσι μεταφέρεται όλη η δομή του `Position`, μαζί με την τρέχουσα βαθμολογία και το χρώμα του παίκτη που έχει σειρά να παίξει. Το μήνυμα έχει τη μορφή `4 4 4 2 1 1 0 ... 2 2 4 4 4 11 8 1` για ένα μικρό ταμπλώ ακτίνας/πλευράς τεσσάρων κυψελών, όπου τα πρώτα `ARRAY_BOARD_SIZE*ARRAY_BOARD_SIZE` bytes είναι τα περιεχόμενα του ταμπλώ μας, τα επόμενα δύο η τρέχουσα βαθμολογία του λευκού και μαύρου παίκτη αντίστοιχα και τέλος η ταυτότητα του παίκτη που έχει την κίνηση στην τρέχουσα κατάσταση.

- `NM_COLOR_W`
  Ο server μάς ενημερώνει ότι το χρώμα μας από εδώ και στο εξής είναι το λευκό. Το μόνο που έχουμε να κάνουμε είναι να ενημερώσουμε τον παίκτη μας γι' αυτήν την αλλαγή. Στον έτοιμο κώδικα αυτό αποθηκεύεται στη μεταβλητή `myColor` μέσα στο `client.c`.

- `NM_COLOR_B`
  Αντίστοιχα με το παραπάνω μήνυμα, αλλάζουμε το χρώμα του παίκτη μας σε μαύρο.

- `NM_REQUEST_MOVE`
  Ο server ζητάει την κίνησή μας. Πρέπει να αποφασίσουμε ποια θα είναι αυτή και αφού γεμίσουμε τη δομή της κίνησης, να την αποστείλουμε με την βοήθεια της συνάρτησης `void sendMove( Move * moveToSend, int mySocket )`. Στην ουσία, μεταφέρεται ο μονοδιάστατος πίνακας `tile` της δομής `Move` με τις συντεταγμένες του σημείου που επιλέξαμε. Αν θέλουμε να στείλουμε την κίνηση $(5, 0)$, το μήνυμα θα έχει τη μορφή `5 0` και τίποτα παραπάνω. Το χρώμα του παίκτη μας δεν αποστέλλεται, καθώς ο server ξέρει πολύ καλά ποιοι είμαστε! Προφανώς, δεν μπορείτε να στείλετε illegal move ως δήθεν αντίπαλος...! Στη συνέχεια, πριν επεξεργαστούμε το επόμενο μήνυμα του server, θα πρέπει να ενημερώσουμε τη σκακιέρα με την κίνησή μας.

- `NM_PREPARE_TO_RECEIVE_MOVE`
  Ο server μας ενημερώνει ότι ο αντίπαλός μας έχει επιλέξει την κίνησή του και τη λαμβάνουμε εκτελώντας τη συνάρτηση `void getMove( Move * moveToGet, int mySocket )`. Στη συνέχεια, πριν αναζητήσουμε το επόμενο μήνυμα, πρέπει να παίξουμε αυτήν την κίνηση με το αντίπαλο χρώμα στο ταμπλώ μας για να το ενημερώσουμε.

- `NM_REQUEST_NAME`
  Ο server θέλει να μάθει το όνομά μας. Αυτό αποθηκεύεται στο string `agentName` μέσα στο `client.c`. Βάλτε το όνομα που επιθυμείτε, έχοντας υπόψη τον περιορισμό των `MAX_NAME_LENGTH` χαρακτήρων (8 στην περίπτωσή μας). Η αποστολή γίνεται με χρήση της `void sendName( char textToSend[ MAX_NAME_LENGTH +1 ], int mySocket )`. Το μήνυμα που αποστέλλεται έχει μπροστά ένα byte που δηλώνει το μέγεθος του ονόματος. Για παράδειγμα, αν θέλουμε να στείλουμε το `test`, το μήνυμά μας θα είναι `4 t e s t`.

- `NM_QUIT`
  Ο server μας ζητάει να αποσυνδεθούμε και να αναστείλουμε τη λειτουργία μας, συνεπώς θα κλείσουμε το socket της επικοινωνίας και θα τερματίσουμε.

Μετά το πέρας της επεξεργασίας κάθε μηνύματος (εκτός από αυτό του τερματισμού), περιμένουμε νέο μήνυμα από τον server για να συνεχίσουμε.

5.4 Client

Ο client είναι σχεδιασμένος για να συνδέεται στον server και να παίζει το ρόλο ενός παίκτη. Ο server είναι αυτός που αποφασίζει ποια πλευρά θα παίξει ο κάθε ένας client. Για να τρέξει ο C client απλά γράφουμε:

`./client [-i server-ip] [-p server-port]`.

- `server-ip`: ορίζει την ip address του μηχανήματος στο οποίο τρέχει ο server, π.χ. `147.27.14.5`, (default=`127.0.0.1` – το ίδιο μηχάνημα με τον client).
- `server-port`: ορίζει το port στο οποίο θα συνδεθεί ο client (default=`6002`).

5.5 Server

Το γραφικό περιβάλλον guiServer (`./guiServer32` ή `./guiServer64`) που δίνεται επιτρέπει τη διεξαγωγή παρτίδων HexThello μεταξύ ενός ενσωματωμένου παίκτη που παίζει εντελώς τυχαία, διαφόρων εξωτερικών παικτών που μπορούν να συνδεθούν μέσω δικτύου, και χρηστών (ανθρώπων). Ο χρήστης έχει τη δυνατότητα να αλλάξει τους παίκτες κατά την διάρκεια μιας παρτίδας, χρησιμοποιώντας οποιεσδήποτε από τις διαθέσιμες επιλογές παικτών. Επίσης, μπορεί να αποσυνδέσει και να συνδέσει όποιον εξωτερικό παίκτη θέλει. Οπτικά, ο χρήστης υποβοηθείται με το να επισημαίνονται με γκρίζο χρώμα πάνω στο ταμπλώ οι έγκυρες διαθέσιμες κινήσεις και με κόκκινο κύκλο η τελευταία κίνηση που έγινε. Εναλλακτικά, για την διεξαγωγή μιας παρτίδας μπορεί να χρησιμοποιηθεί ο ascii server (`./server`).

6 Επίλογος

Όπως θα έχετε διαπιστώσει, η εργασία είναι αρκετά ανοικτή με την έννοια ότι σας δίνει την ελευθερία να κάνετε πολλές επιλογές χωρίς ιδιαίτερους περιορισμούς και να επικεντρωθείτε στα σημεία που σας ενδιαφέρουν περισσότερο. Επίσης, σας δίνει τη δυνατότητα για μελλοντικές επεκτάσεις με μεθόδους μηχανικής μάθησης, ώστε ο παίκτης να βελτιώνει την απόδοσή του σταδιακά. Ξεκινήστε σήμερα κιόλας, βάλτε το μυαλό και τη φαντασία σας να δουλέψουν και διασκεδάστε ασκώντας τη δημιουργικότητά σας!

Καλή επιτυχία!!!