ΠΛΗ 417 ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ
2ο project

Παντελιδάκης Μηνάς 2011030110
Θεοδωρής Χαράλαμπος 2011030032

Εισαγωγή :

Στα πλαίσια της εργασίας καλούμαστε να σχεδιάσουμε και να υλοποιήσουμε ένα πρόγραμμα το οποίο θα παίζει το παιχνίδι TUC-CHESS.

Ο τρόπος με τον οποίο θα παίρνονται οι αποφάσεις για την εκπόνηση των κινήσεων είναι μέσω το αλγορίθμου minimax.

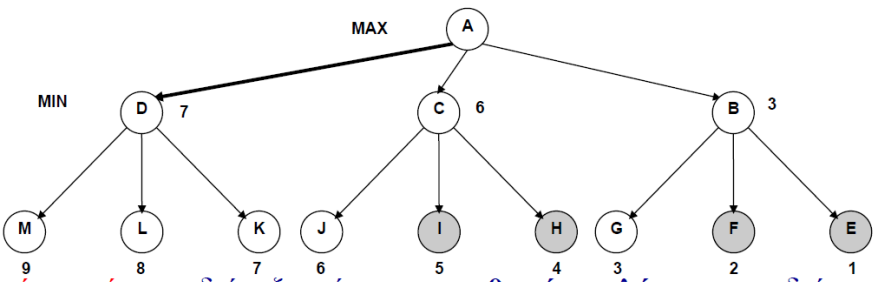
Υλοποίηση :

Χρησιμοποιώντας τα αρχεία κώδικα που μας δόθηκαν στα πλαίσνια του μαθήματος, δημιουργήσναμε σνε γλώσσα Java έναν πράκτορα που τρέχει το παιχνίδι TUC Chess. Η αρχή πίσω από τη λειτουργία του είναι ο αλγόριθμος Minimax, με α-β pruning για μείωσνη της πολυπλοκότητας. Η αξιολόγηση των κινήσεών μας γίνεται στα φύλλα του δέντρου αναδρομής.Η σύνδεσή μας με τον server γίνεται κανονικά και συνεργαζόμαστε χωρίς προβλήματα. Ακολουθούν λιγα λόγια για τις τεχνικές/αλγορίθμους που χρησιμοποήσαμε και υλοποιήσαμε.

Minimax: Ο αλγόριθμος Minimax που χρησιμοποιήσαμε, είναι αυτός που διδαχθήκαμε, με τον Max να είναι ο εκάστοτε παίκτης που καλείται να κινηθεί και ο Min ο ανταγωνισντής του, μέσα στο σενάριο που σκέφτεται ο πρώτος για να αποφασίσει. Το δέντρο του minimax φτιάχνεται για βάθος 5 στην πρώτη κίνηση του παιχνιδιού , καθώς δεν επηρεάζει πολύ την εξέλιξη της παρτίδας , ενώ για τις υπόλοιπες κινήσεις ο minimax φτάχνει το δέντρο για βάθος 8.

α-β Pruning: Γίνεται ακριβώς όπως στον ψευδοκώδικα του βιβλίου με στόχο τη μείωση των άσκοπων εξερευνήσεων.

Move ordering : Για να πετύχουμε ακόμα μεγαλύτερο κλάδεμα του δέντρου παιχνιδιού, οι απόγονοι κάθε κόμβου max διατάσσονται από τον καλύτερο προς τον χειρότερο, ενώ οι απόγονοι κάθε min κόμβου από τον χειρότερο (για τον max) προς τον καλύτερο. Αυτή είναι πλέον η σειρά με την οποία θα εκεταστούν.



ευρετικός κανόνας: «διάταξη» ώστε τα **«πιθανώς καλύτερα»** παιδιά να εξετάζονται πρώτα...

Cutoff function : Δεν υπάρχει, αν και θα έπρεπε να γίνεται βάσει του στιγμιοτύπου της σκακιέρας που πήραμε από το server. Η αποκοπή και το evaluation γίνονται για hard coded depth = 8 στη γενική περίπτωση.Φυσιολογικά θα έπρεπε να μεταβάλλεται με βάση τα εναπομείναντα πιόνια και της σημαντικότητάς τους.

Evaluation function : Εδώ οι ιδέες ήταν πολλές αλλά ο χρόνος λίγος. Η αλήθεια είναι πως δεν προλάβαμε να πειραματιστούμε αρκετά. Η συνάρτηση που υλοποιήσαμε παίρνει ένα συσσωρευτικό score για τον maximizer και ένα για τον minimizer. Αυτά τα scores προκύπτουν από τα κέρδη των 2 παιχτών σε κάθε κόμβο του δέντρου παιχνιδιού. Τα κέρδη αυτά δεν υπολογίζονται έξυπνα. Είναι η άμεση επίδραση της εκάστοτε κίνησης του παίχτη, δηλαδή +1 για αιχμαλώτιση στρατιώτη, +3 για αιχμαλώτιση πύργου και +7 για αιχμαλώτηση βασιλιά. Αφού υπολογιστούν αυτά για όλες τις κινήσεις που έγιναν μέχρι να φτάσουμε στα φύλλα του δέντρου παιχνιδιού (όριο αποκοπής) η χρησιμότητα της κίνησης προκύπτει ως εξής:

```
if(initColor==0) {
        value = maxiMizerScore - miniMizerScore + whitePieces - blackPieces;
        if(whiteKingIsUp){value+=5;}
        if(blackKingIsUp){value-=5;}
        value += wRook -bRook;
} else {
        value = maxiMizerScore - miniMizerScore - whitePieces + blackPieces;
        if(whiteKingIsUp){value-=5;}
        if(blackKingIsUp){value+=5;}
        value += bRook -wRook;
}
```

initColor είναι το αρχικό χρώμα που δόθηκε στον client. 0 έχει ο άσπρος , 1 ο μαύρος
blackRook και whiteRook έχουν τιμή ++3 για κάθε πύργο που υπάρχει ακόμα στο board .

Η αξία κίνησης ενός στρατιώτη κατά μία θέση μπροστά (ακόμα και αν δεν τρώει τίποτα) είναι 0.1 για να ενθαρρύνουμε τους στρατιώτες να προχωρήσουν.

Για να βελτιώσουμε περαιτέρω την απόδοση του αλγορίθμου minimax σκεφτήκαμε να υλοποιήσουμε transposition table , αλλά δυστυχώς δεν προλάβαμε.

Πρόκειται για το caching ενεργειών και αποτελεσμάτων (γνωρίζοντας εκ των προτέρων πως η εκάστοτε ενέργεια έχει το Χ αποτέλεσμα).
Για παράδειγμα αν οι παίχτες κάνουν 2 κινήσεις που καταλήγουν στο ίδιο ακριβώς board με διαφορετική σειρά έχουμε 4 δυνατά transposition boards.

Singular extensions και forward prunning δεν υλοποιήθηκαν.

Τέλος , ο client συνεργάζεται με το server χωρίς προβλήματα.