ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ
ΠΛΗ417

LAB41740898

2η Εργασία Προγραμματισμού

Φλέγγας Γεώργιος 2014030161

Θεοδωράκη Εμμανουέλα 2014030238

02/06/19

ΗΜΜΥ

ClientM για το παιχνίδι TUC-CHESS

Εισαγωγή

Στα πλαίσια της 2ης εργασίας προγραμματισμού σχεδιάσαμε και υλοποιήσαμε ένα πρόγραμμα-client, το οποίο παίζει το παιχνίδι TUC-CHESS. Το παιχνίδι αυτό, αποτελεί μια παραλλαγή του κλασσικού παιχνιδιού σκάκι, που δημιουργήθηκε για τις ανάγκες της εργασίας αυτής. Στο παιχνίδι αυτό θα έρχονται αντιμέτωποι clients μέσα από έναν κεντρικό server και θα περιλαμβάνει ειδικά γραφικά παράθυρα που θα εμφανίζουν τις κινήσεις πάνω στην σκακιέρα, καθώς και τα μηνύματα σχετικά με την εξέλιξη του παιχνιδιού που θα λαμβάνει και θα στέλνει ο server από και προς τους χρήστες.

TUC-CHESS

Κανόνες παιχνιδιού:

Το παιχνίδι παίζεται σε μία σκακιέρα διαστάσεων 7 × 5 και υπάρχουν δύο παίκτες, ο λευκός και ο μαύρος, καθένας από τους οποίους έχει 7 πιόνια, 2 πύργους και 1 βασιλιά. Οι παίκτες παίζουν εναλλάξ και, κατά σύμβαση, η πρώτη κίνηση ανήκει στο λευκό παίκτη.

Υπάρχουν 3 διαφορετικά είδη πεσσών: α) απλά πιόνια, β)πύργοι, γ)βασιλιάδες. Τα πιόνια μετακινούνται πάντα προς τη μεριά του αντιπάλου, είτε ένα βήμα μπροστά αν δεν υπάρχει κάποιος πεσσός σε εκείνη τη θέση είτε ένα βήμα διαγωνίως (αριστερά ή δεξιά) αν υπάρχει κάποιος πεσσός του αντιπάλου σε εκείνη τη θέση. Οι πύργοι μετακινούνται μέχρι και κατά τρεις θέσεις προς οποιαδήποτε κάθετη και οριζόντια κατεύθυνση (μη διαγώνια). Αν στις ενδιάμεσες θέσεις κάποιας κατεύθυνσης υπάρχει πεσσός τότε δεν επιτρέπεται η μετακίνηση. Οι βασιλιάδες μετακινούνται προς οποιαδήποτε κάθετη και οριζόντια κατεύθυνση κατά μία θέση.

Στο τέλος του παιχνιδιού ανακηρύσσεται νικητής ο παίχτης που έχει την μεγαλύτερη βαθμολογία. Κάθε παρτίδα λήγει είτε όταν ένας από τους δύο βασιλιάδες αιχμαλωτιστεί ή όταν στο παιχνίδι δεν έχει μείνει κανένας άλλος πεσσός εκτός των δύο βασιλιάδων ή όταν ξεπεραστεί το χρονικό όριο των 12 λεπτών. Υπάρχουν 3 διαφορετικοί τρόποι για να συλλέξει ένας παίχτης βαθμούς: 1) συλλογή bonus, 2) αιχμαλωσία αντίπαλου πεσσού, 3) να φτάσει ένα πιόνι στη τελευταία γραμμή. Ένα bonus δίνει, στον παίκτη που μετακινείται στη θέση που αυτό βρίσκεται, 1 πόντο με πιθανότητα 0.85 και 0 πόντους με πιθανότητα 0.15. Ένα bonus εμφανίζεται τυχαία μετά από την κίνηση ενός παίχτη σε κενές θέσεις με πιθανότητα 0.25. Η αιχμαλώτιση ενός πιονιού, πύργου, βασιλιά αυξάνει τη βαθμολογία του παίκτη κατά 1,3,8 πόντους αντίστοιχα.

Επικοινωνία με server:

Αρχικά ο server περιμένει τους 2 παίχτες να συνδεθούν και τους αναθέτει με σειρά προτεραιότητας ως **PW** και **PB**. Στην συνέχεια στέλνει το μήνυμα **GB**(game begin) και περιμένει την κίνηση του player white. Για να αποφασίσει την κίνηση του, ο κάθε παίχτης έχει 4 δευτερόλεπτα στην διάθεση του. Στην συνέχει πρέπει να την στείλει στον server στην μορφή: *"ABCD"* ισχύει ότι A, C ∈ {0, 1, 2, 3, 4, 5, 6} (7 γραμμές) και B, D ∈ {0, 1, 2, 3, 4} (5 στήλες). Όταν λάβει την κίνηση αυτή, ελέγχει ότι είναι έγκυρη, την εκτελεί στην σκακιέρα και στέλνει στους 2 παίχτες μήνυμα της μορφής "*TABCDEFGHIJK*", όπου A δηλώνει τον επόμενο παίχτη που θα παίξει, BCDE την κίνηση που μόλις πραγματοποιήθηκε, FG την θέση του καινούργιου bonus που εμφανίστηκε και HIJK την τρέχουσα βαθμολογία. Η ανταλλαγή αυτών των μηνυμάτων θα συνεχιστεί μέχρι να τελειώσει το παιχνίδι και να στείλει ο server το μήνυμα "*GEABCD*", όπου GE δηλώνει το game end και το ABCD το τελικό score.

ClientM

Βασική ιδέα:

Για την ανάπτυξη του τρόπου λήψης απόφασης σχετικά με την βέλτιστη κίνηση που θα εκτελεί ο client μας, μελετήσαμε τους αλγόριθμους minimax και alpha-beta pruning. Αξιοποιώντας τις ιδέες πάνω στις οποίες βασίζονται οι αλγόριθμοι αυτοί, υλοποιήσαμε μια απλή εκδοχή τους. Στόχος του αλγορίθμου μας είναι να διαλέξει την κίνηση, η οποία θα μεγιστοποιεί τα οφέλη μας και περιορίζει αυτά του αντιπάλου. Η βασική του δομή είναι η εξής:

```
ourmove:
      score=-100
      copy board into temp_board
      copy availableMoves into av_moves
      for(i=0;i<av_moves.size();i++)
            restore board from temp_board
            clear availableMoves
            action=av_moves.get(i)
            decode action into x1,y1,x2,y2
            calculate my_score gain for this action
            make this move
            find availableMoves for enemy player
            new_score=best_move()
            new_score=Evaluation(my_score,new_score)
            if(new_score>score)
                  score=new_score
                  save this move as best
            else if(new_score=score)
                  if((board[best_row][best_col]="" ||board[best_row][best_col]="P")
                   &&(board[x2][y2]=enemy pawn)
                        set this move as new best
      restore board from temp_board
      return best
```

Η κλήση της συνάρτησης αυτής γίνεται αφού έχει καλεστεί μια εκ των blackMoves() ή whiteMoves() και άρα η arraylist availableMoves, περιλαμβάνει τις διαθέσιμες κινήσεις για τον παίχτη μας. Επειδή όμως εμείς αργότερα θα καλέσουμε μια εκ των 2 για να μας δώσει τις κινήσει του αντιπάλου, αφού έχουμε εκτελέσει κάποια από τις πιθανές κινήσεις πάνω στο board, πρέπει να αποθηκεύσουμε σε προσωρινά αντίγραφα το availableMoves και το board. Στην συνέχεια ξεκινάμε να εκτελούμε μια μια τις διαθέσιμες κινήσεις του παίχτη μας και υπολογίζουμε το όφελος που θα είχε ο παίχτης μας βαθμολογικά, εάν πραγματοποιούσε της κίνηση αυτή,το αποθηκεύουμε μέσα στην μεταβλητή my_score και εκτελούμε της κίνηση πάνω στο board. Αφού εκτελέσουμε την κίνηση, καλούμε μια εκ των blackMoves() ή whiteMoves(), ώστε να αποκτήσεουμε τις διαθέσιμες κινήσεις του αντιπάλου. Στην συνέχεια, βρίσκουμε την κίνηση του αντιπάλου, δεδομένο της κίνησης μας, η οποία θα του επιφέρει το καλύτερο δυνατό score μέσω της συνάρτησης best_move() και το αποθηκεύουμε στην new_score .

Καλώντας την συνάρτηση Evaluation(my_score,new_score) υπολογίζουμε πόσο θα είναι εν τέλη το new_score, ώστε να το συγκρίνουμε με αυτό της καλύτερης μέχρι τώρα κίνησης και να αποφασίσουμε, αν η κίνηση που εξετάζουμε τώρα είναι καλύτερη από την μέχρι τώρα best move.(Το score έχει αρχικοποιηθεί ως -100 και έτσι η πρώτη κίνηση ορίζεται ως best). Ακόμα αν βρούμε μια κίνηση η οποία έχει ίδιο score με την μέχρι τώρα καλύτερη, τότε ελέγχουμε αν η προηγούμενη best move, θα μετακινούσε πεσσό σε θέση που ήταν κενή ή υπήρχε bonus και αν η κίνηση που εξετάζουμε εκείνη την στιγμή, θα οδηγήσει στην αφαίρεση αντίπαλου πεσσού από το παιχνίδι. Σε περίπτωση που ισχύουν οι συνθήκες αυτές θέτουμε την νέα κίνηση ως best move, αφού το να αφαιρέσουμε ένα πιόνι του αντιπάλου μας οδηγεί σε πιο πλεονεκτική θέση σε σχέση με το να κάνουμε μια απλή κίνηση ή να πάρουμε ένα bonus.

Εν συνεχεία, επαναφέρει το board στην κατάσταση πριν εκτελέσει την κίνηση και δοκιμάζει την επόμενη διαθέσιμη μέχρι να μην έχει άλλη. Εν τέλει, επαναφέρει το board για μια τελευταία φορά και στέλνει στον server την καλύτερη κίνηση που διάλεξε.

Evaluation function:

Για να γίνει η σύγκριση μεταξύ της αποτελεσματικότητας της κάθε κίνησης ήταν απαραίτητη η υλοποίηση μιας καλής συνάρτησης αξιολόγησης.

Ξεκινήσαμε από την πιο απλή εκδοχή και βήμα-βήμα και μέσω διαφόρων πειραμάτων που τρέξαμε (π.χ. ClientM vs RandomMove, ClientM vs ClientM) την εξελίξαμε μέχρι να έχουμε μια αρκετά ικανοποιητική. Η αρχική μας αποτελούσε την απλή σύγκριση μεταξύ του score που θα κερδίσει ο παίχτης μας με την κίνηση αυτή, μείον το καλύτερο score που θα έχει ο αντίπαλος ως απάντηση: Eval= myscore-newscore . Στην συνέχεια αντιληφθήκαμε την αξία του να αφαιρούμε με την κίνηση μας πεσσούς του αντιπάλου, οπότε συμπεριλάβαμε τον αριθμό των διαθέσιμων πεσσών του κάθε παίχτη μετά την κίνηση στην συνάρτηση μας :

Eval=(myscore+mypawncount)-(newscore+enemypawncount).

Όμως σημαντικό είναι το γεγονός ότι κάθε είδος πεσσού, έχει διαφορετικό score. Για αυτόν τον λόγο αντικαταστήσαμε το pawncount με το pawnvalue. Το pawnvalue αποτελεί το άθροισμα της αξίας του κάθε είδους πεσσού επί τον αριθμό των πεσσών αυτών που βρίσκονται στην σκακιέρα:

pawnvalue=(NumOfpawn * 1)+(NumOfrook * 3)+(NumOfking * 8)

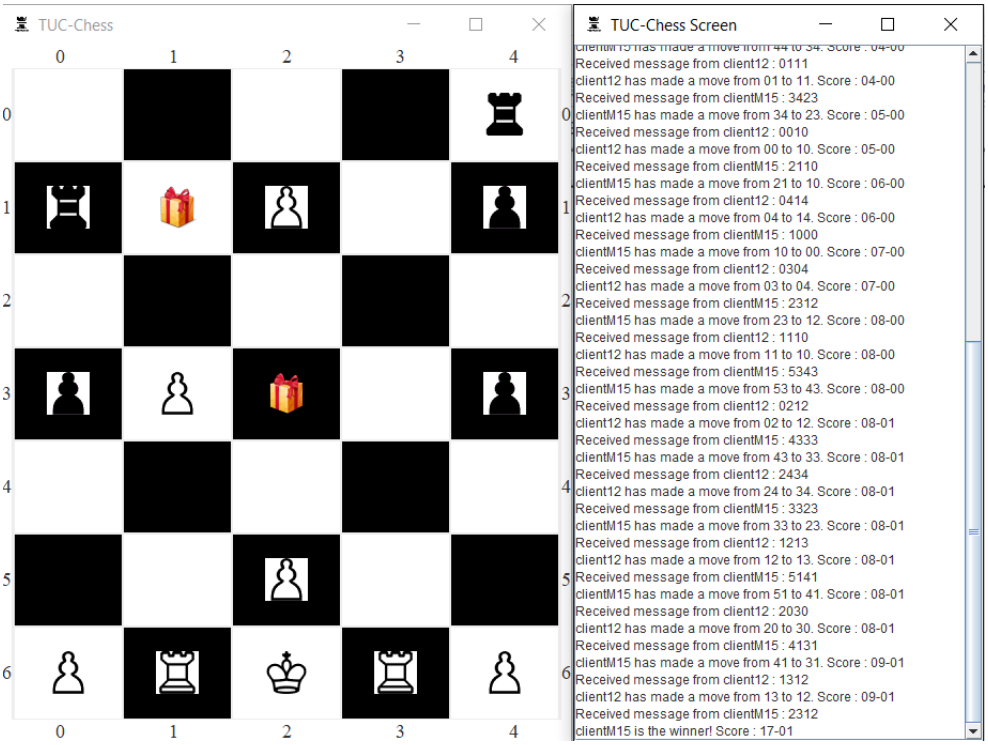
Άρα η τελική μορφή της συνάρτησης αξιολόγησης μας είναι:

Eval = (myscore + mypawnvalue) - (newscore + enemypawnvalue)

Αποτελέσματα

Ακολουθούν 2 τελικά αποτελέσματα παιχνιδιών που τρέξαμε, ώστε να επιβεβαιώσουμε την σωστή λειτουργία του αλγορίθμου μας:

clientM: Whiteplayer vs client: Blackplayer



clientM16: Whiteplayer vs clientM7: Blackplayer