Τεχνητή Νοημοσύνη Εργασία 1

Μέλη ομάδας :	ΑΜ :
Δημήτριος Τσιομπίκας	3180223
Νικόλαος Χριστοδούλου	3180206
Παναγιώτης Παναγιώτου	3180139

Περίληψη

Αρχικά , η εργασία που επιλέξαμε ήταν το Reversi/Othello και την υλοποιήσαμε προσαρμόζοντας τον κώδικα του εργαστηρίου 4 ώστε να υποστηρίζει Reversi αντί για τρίλιζα. Η επιλογή κινήσεων γίνεται με τη χρήση του αλγόριθμου Minimax με a-b πριόνισμα και το AI προσπαθεί να κερδίσει τον παίκτη παίζοντας με τις βέλτιστες κινήσεις. Η Move Class δεν έχει υποστεί αλλαγές.

Οι αλλαγές που έχουμε κάνει είναι οι εξής :

GamePlayer Class

Στην Gameplayer αλλάξαμε το default Constructor ώστε να βάζει απευθείας Black color στα πούλια , στη μέθοδο MiniMax() επειδή βάσει κανόνων ο παίκτης με μαύρα πούλια

ξεκινά πρώτος θα είναι ο max στο δέντρο μας ενώ ο παίκτης με τα λευκά πούλια θα είναι ο min.

Board Class

Αυτή η κλάση έχει υποστεί τις περισσότερες αλλαγές και θα τις εξηγήσουμε αναλυτικά και όπου χρειάζεται κατά μέθοδο :

Έχουμε αλλάξει τις αρχικές μεταβλητές σε W,B για κάθε χρώμα αντίστοιχα και τον default constructor έτσι ώστε να δημιουργεί απευθείας ένα 8x8 πίνακα τοποθετώντας και τα αρχικά πιόνια στο κέντρο του πίνακα.

Μέθοδος Flip : η Flip δημιουργήθηκε για να μπορεί ο παίκτης και το AI να παίρνουν τα πούλια ο ένας του άλλου. Έχουμε βάλει τα ορισμάτα row , col που είναι για γραμμές , στήλες αντίστοιχα και τα now,before όπου now ο παίκτης που παίζει τώρα και before ο παίκτης που έπαιξε πριν. Ελέγχουμε για το κελί όπου μπορεί να γίνει έγκυρη κίνηση σε ποια από τις 8 κατευθύνσεις μπορεί να αλλάξει τα αντίπαλα πούλια.

Μέθοδος isValidMove : η isValidMove προσαρμόστηκε στις ανάγκες του Othello ως εξής : έχουμε πάλι τα ορίσματα row,col και προσθέσαμε τα now,before. Εδώ θα μπορούσαμε να μην βάλουμε τα ορίσματα now,before και να χρησιμοποιήσουμε τη

μεταβλητή lastLetterPlayed αλλά λόγω της μορφής του κώδικα θα έπρεπε να ελέγξουμε 16 διαφορετικές περιπτώσεις (8 για τον Black pawn player 8 για τον White pawn player) ενώ τώρα ελέγχουμε μόνο 8. Το ίδιο ισχύει και για την Flip. Εδώ κάνουμε τους εξής ελέγχους : η κίνηση πρέπει να γίνεται εντός ορίων του πίνακα , το κελί που θα γίνει η κίνηση να είναι άδειο (να μην περιέχει γράμμα W ή B) και να ελέγχει ΟΛΑ τα γειτονικά κελιά από το κελί που θέλουμε να κάνουμε κίνηση.

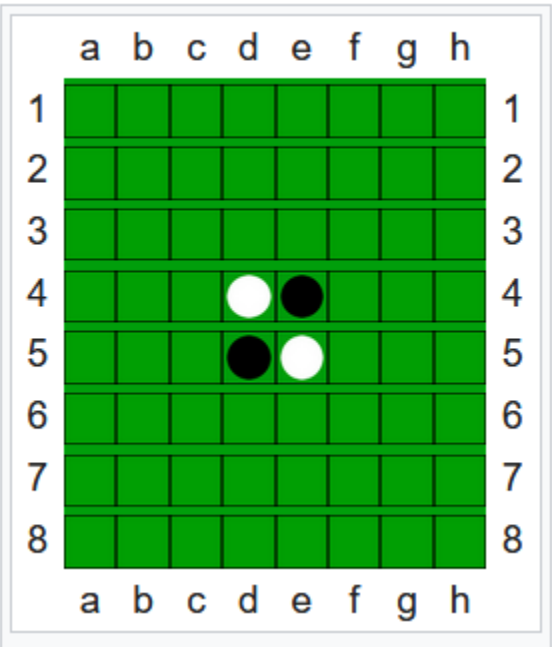
Μέθοδος Available : Η μέθοδος αυτή παίρνει σαν ορίσματα ένα αντικείμενο Board , τον παίκτη που παίζει μετά και τον παίκτη που παίζει τώρα. Ουσιαστικά , ελέγχει αν ο παίκτης που παίζει τώρα έχει διαθέσιμες κινήσεις.

Μέθοδος evaluate : η evaluate είναι η ευρετική μας , η οποία λειτουργεί ως εξής : Βρίσκει το πλήθος από πιόνια που έχει ο κάθε παίκτης και βρίσκει τη διαφορά τους. Βάσει αυτής της ευρετικής το AI αποφασίζει μαζί με τον αλγόριθμο minimax την βέλτιστη κίνηση που θα παίξει.

Μέθοδος Terminal : η Terminal προσαρμόστηκε ώστε να παίρνει ένα όρισμα τύπου ακέραιας μεταβλητής (gameover) ώστε όταν το gameover ισούται με 2 η Terminal να επιστρέφει true , Να μας δείχνει τα σκορ των παικτών, ποιος νίκησε και το παιχνίδι να τερματίζει (Το gameover γίνεται 2 όταν κανένας

παίκτης δεν έχει άλλη έγκυρη κίνηση. Αυτό μας καλύπτει και στην περίπτωση που το board γεμίσει από πούλια καθώς και πάλι οι παίκτες ΔΕΝ έχουν άλλη έγκυρη κίνηση).

Μέθοδος Print : Τέλος , η Print προσαρμόστηκε έτσι ώστε να μας βγάζει το board του Reversi όπως αυτός παρουσιάζεται στον σύνδεσμο της Wikipedia για το Reversi.



Main Class

Η Main class είναι η κλάση που εκτελεί το game. Εδώ ζητάμε βάθος αναζήτησης για τον Minimax από τον χρήστη και αν

θέλει να παίξει πρώτος ή δεύτερος. Τα επίπεδα κρίνονται σε εύκολο, μέτριο , δύσκολο ως εξής :

Depth = 2 (εύκολο)

Depth = 4 (μέτριο)

Depth = 6 (δύσκολο)

(Το βάθος μπορεί να αυξηθεί και παραπάνω με την επίπτωση ότι θα παίρνει παραπάνω χρόνο να αποφασίσει το AI για την επόμενη κίνησή του καθώς θα ψάχνει σε μεγαλύτερο βάθος)

Στη συνέχεια αναθέτονται τα χρώματα στον παίκτη και το AI ανάλογα με τη σειρά που επέλεξε να παίξει. Εδώ ελέγχουμε τη μεταβλητή gameover η οποία υποδηλώνει ποιος παίκτης δεν έχει διαθέσιμες κινήσεις. Ύστερα υλοποιούμε την περίπτωση που παίζει ο Human player , λέγοντάς του να παίξει μία κίνηση.

Η κίνηση γίνεται ως εξής : βάζουμε τη γραμμή και τη στήλη όπως είναι στο board μας , πχ αν θέλουμε να παίξουμε στη θέση (0,0) θα βάλουμε στα inputs row : 1 col : A (το πρόγραμμά μας δέχεται και μικρούς χαρακτήρες και κεφαλαίους πχ A,a).

Αν δεν έχει διαθέσιμες κινήσεις δίνεται η σειρά στο AI και σε κάθε κίνηση που γίνεται εμφανίζεται το board με την νέα κίνηση.

Αμέσως μετά ακολουθεί η περίπτωση που παίζει το AI , το οποίο χρησιμοποιεί τον αλγόριθμο Minimax για να κάνει την κίνησή του. Στην σπάνια περίπτωση που ο Minimax δεν

καταφέρει να βρει έγκυρη κίνηση , έχουμε υλοποιήσει ειδικό κώδικα που λύνει το πρόβλημά μας παίζοντας την πρώτη διαθέσιμη κίνηση για το AI που βρίσκει. Ισχύουν οι ίδιοι περιορισμοί εδώ , ότι αν ο παίκτης καταφέρει να φέρει το AI σε αδιέξοδο (να μην έχει διαθέσιμες κινήσεις να παίξει) δίνεται η σειρά στον Human player. Τέλος, αν το gameover = 2, δηλαδή τελειώσουν οι διαθέσιμες κινήσεις για τον Human player και το AI τότε το παιχνίδι τελειώνει και εκτελείται η συνάρτηση Terminal όπου μας βγάζει το σκορ και τον νικητή.

Μερικά πειράματα που κάναμε ήταν τα εξής : παίξαμε με το AI σε βάθος = 6 , παίζοντας ως White pawn player και το σκορ βγήκε :

Black : 43

White : 21

Όπου το AI πήρε την νίκη και επίσης τεστάραμε το medium level με βάθος = 4 παίζοντας ως Black pawn player αυτή τη φορά με τελικό σκορ :

Black : 27

White : 37

Όπου το AI πάλι κατάφερε να κερδίσει αλλά με μικρότερη διαφορά από πριν καθώς έψαχνε σε μικρότερο βάθος.

Τέλος, κάναμε δοκιμή και με το εύκολο επίπεδο (depth = 2) , παίζοντας ως black pawn player με τελικό σκορ :

Black : 37

White : 27

Όπου νικήσαμε εμείς με διαφορά 10 πόντων , καθώς το AI έψαχνε σε πολύ μικρό βάθος και δεν έκανε πάντα την καλύτερη δυνατή κίνηση.