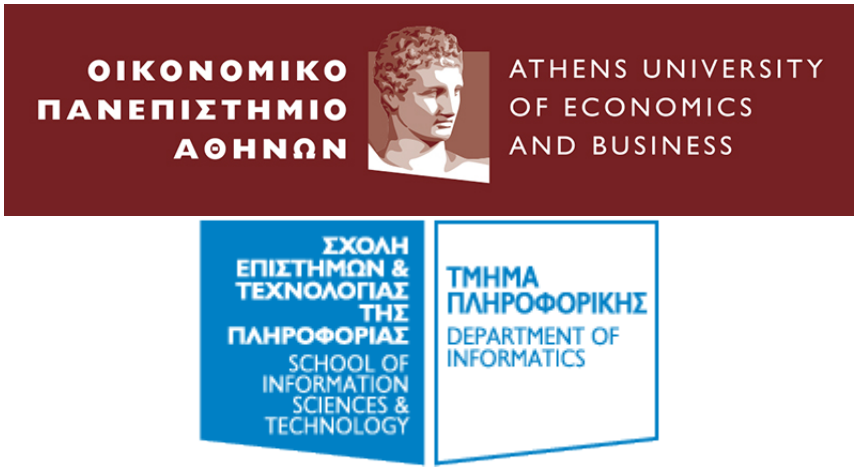


1η ΕΡΓΑΣΙΑ ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ

ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2020

ΦΩΤΙΟΣ ΠΑΝΟΣ -3180141
ΙΩΑΝΝΗΣ ΠΑΠΑΧΡΗΣΤΟΥ - 3180150

A MINUTE TO LEARN...A LIFETIME TO MASTER!

Περιεχόμενα

1 Εισαγωγή

Στα πλαίσια της πρώτης προγραμματιστικής εργασίας του μαθήματος της Τεχνητής Νοημοσύνης, κληθήκαμε να υλοποιήσουμε μια εφαρμογή για το γνωστό επιτραπέζιο παιχνίδι *Othello®*, χρησιμοποιώντας τον αλγόριθμο *Minimax* για τις κινήσεις του CPU παίκτη. Η υλοποίηση έγινε με χρήση της γλώσσας προγραμματισμού *Java* και περισσότερες πληροφορίες σχετικά με τον τρόπο εγκατάστασης, μεταγλώττισης και εκτέλεσης μπορείτε να βρείτε στο αρχείο README.txt που θα βρείτε στον φάκελο του κώδικα.

2 Κλάσεις

Για την καλύτερη κατανόηση του τρόπου λειτουργίας του προγράμματος μας, θα αναλύσουμε κάθε κλάση ξεχωρίστα, εξηγώντας την ιδέα πίσω από αυτές και τον λόγο ύπαρξής τους καθώς και τον τρόπο υλοποίησής τους.

2.1 Η κλάση Color

Αρχίζοντας από την απλούστερη, η κλάση Color περιέχει τέσσερις στατικές μεταβλητές που χρησιμοποιήσαμε έτσι ώστε να της επικαλούμαστε όταν θέλουμε να αναφερθούμε σε κάποιο "χρώμα" του παιχνιδιού προς αποφυγή σύγχησης. Έτσι, τα τέσσερα χρώματα που ορίσαμε (Empty, Black, White, Star) ατιστοιχούν στις περιπτώσεις Κενό Κελί, Μαύρο, Λευκό και Επιτρεπτή Θέση. Διευκρινίζουμε πως το μαύρο και το λευκό δεν αξιοποιούνται μόνο για την διάκριση των χρωμάτων των κελιών αλλά και για τους παίκτες που παίζουν, σε αντίθεση με τα άλλα δύο που αναφέρονται μόνο σε κενά κελιά και κενά κελιά που μπορεί ο τρέχον παίκτης να τοποθετήσει ένα πιόνι αντίστοιχα.

2.2 Η κλάση Square

Ένα αντικείμενο τύπου Square αντιπροσωπεύει μια ψηφίδα, ένα τετραγωνάκι δηλαδή από το ταμπλό του παιχνιδιού. Χαρακτηριστικά του είναι οι συντεταγμένες του στον πίνακα του παιχνιδιου, x και y, και το χρώμα του τετραγώνου, κενό, μαύρο, λευκό ή πιθανή θέση. Χάρην ευκολίας δεν δημιουργήσαμε κλάση Πιόνι αφού αξιοποιήσαμε τον χρωματισμό των τετραγώνων για την λειτουργία αυτή.

2.2.1 Constructors - Getters - Setters

Ο πρώτος κατασκευαστής (Square(int x, int y)) αξιοποείται για την αρχικοποίηση ενός κελιού του πίνακα που αρχικά είναι κενό. Ο δεύτερος (Square(int x, int y, int color)) αρχικοποιεί ένα κελί αλλά του προσδίδει και το χρώμα που δίδεται σαν όρισμα, όπως στην περίπτωση των κελιών [4,4][4,5][5,4][5,5] που στην αρχή του παιχνιδιού έχουν ήδη ένα χρώμα. Τέλος οι μέθοδοι getColor, setColor, getX getY αξιοποιούνται για την ανάγνωση και την εγγραφή δεδομένων των χαρακτηριστικών του κάθε Square αντικειμένου εκτός της κλάσης.

2.2.2 Η μέθοδος changeColor

Η μέθοδος αυτή αξιοποείται για την εναλλαγή του χρώματος των πλακιδιών όταν ένας παίκτης κάνει την κίνησή του. Καλείται εντός της μεθόδου play της κλάσης Board.

2.2.3 Η μέθοδος getContent

Η μέθοδος αυτή μετατρέπει το χρώμα που βρίσκεται αποθηκευμένο σε κάθε τετράγωνο σε σύμβολο ώστε να δημιουργηθεί μια κατανοητή απεικόνιση του πίνακα του παιχνιδιού ως διεπαφή χρήστη.

2.3 Η κλάση Move

Τα αντικείμενα τύπου Move αντιπροσωπεύουν τις κινήσεις που μπορούν να γίνουν στο ταμπλό του παιχνιδιού. Περιλαμβάνουν τις συντεταγμένες x και y καθώς και μια λίστα directions. Εκεί αποθηκέυονται τριπλέτες που δείχνουν την κατεύθυνση (row,column $\in [-1, 1]$) προς την οποία τα τετραγωνα πρέπει να αλλάξουν χρώμα, αν η συγκεκριμένη κίνηση παιχθεί, και το πληθος των τετραγώνων που πρέπει αλλάξουν (enemySquares $\in [1, 6]$).

2.3.1 Constructors - Getters - Setters

Ο κατασκευαστής αρχικοποιεί ένα αντικείμενο move με την λίστα των κατεθύνσεων αρχικά κενή. Επιπλέον, υπάρχουν διαθέσιμοι getters και setters για ανάγνωση και τροποποίηση των συντεταγμένων ενός αντικειμένου move καθώς και την ανάγνωση της λίστας των κατευθύνσεων.

2.3.2 Η μέθοδος insertDirection

Η μέθοδος αυτή καλείτε εντός της μεθόδου movesAllowedList της κλάσης Board και εισάγει στην λίστα καταυθύνσεων της κίνησης μια νέα τριπλέτα - πίνακα [row, column, enemySquares].

2.4 Η κλάση Board

Ένα αντικείμενο Board αντιπροσωπεύει το ταμπλό - πίνακα του παιχνιδιού. Αποτελείται από 8×8 Squares και χαρακτηρίζεται από δύο μεταβλητές, την movesLeft που υποδεικνύει πόσα κενά τετράγωνα υπάρχουν ακόμα στον πίνακα και την value που αντιπροσωπεύει την αξία του πίνακα για τον Minimax αλγόριθμο.

2.4.1 Constructors

Υπάρχουν τρεις κατασκευαστές. Κατά περίπτωση, ο πρώτος αρχικοποιεί έναν πίνακα με τα 4 κεντρικά κελιά χρωματισμένα ανάλογα και την τιμή του ίση με 0, ο δεύτερος αρχικοποιεί έναν πίνακα με τα 4 κεντρικά κελιά χρωματισμένα ανάλογα και την τιμή του ίση με την δοθείσα ως όρισμα τιμή και ο τρίτος κλονοποιεί τον πίνακα που δίδεται ως όρισμα.

2.4.2 Η μέθοδος finish

Καλείται σε περίπτωση που θέλουμε να τερματιστεί το παιχνίδι πριν συμπληρω- θούν όλα τα τετράγωνα του πίνακα. *(Π.χ. σε περίπτωση που κανένας από τους δύο παίκτες δεν έχει διαθέσιμη κίνηση)*

2.4.3 Η μέθοδος printBoard

Η μέθοδος αυτή καλείται μετά από κάθε κίνηση επί του πίνακα του παιχνιδιού και δίνει την απεικόνιση του πίνακα και του περιεχόμενου των κελιών του.

2.4.4 Η μέθοδος Score

Ομοίως, καλείται μετά από κάθε κίνηση επί του πίνακα του παιχνιδιού, υπολογίζει το σύνολο των μαύρων και άσπρων τετραγώνων του πίνακα και εκτυπώνει την απεικόνιση της βαθμολογίας και το ποσοστό του πίνακα που απομένει κένο.

2.4.5 Η μέθοδος endOfGame

Πρόκειται για μια ερώτηση λογικής που ελέγχει αν έχει τελειώσει το παιχνίδι, αν δηλαδή τα ελεύθερα τετράγωνα του πίνακα έχουν εκμηδενιστεί. Αν για κάποιο λόγο το παιχνίδι πρέπει να ολοκληρωθεί νωρίτερα, καλείται η finish και θέτει τη movesLeft στο 0 πριν την κλήση της endOfGame.

2.4.6 Η μέθοδος getSquare

Επιστρέφει το Square που βρίσκεται σε συγκεκριμένες συντεταγμένες του πίνακα

2.4.7 Η μέθοδος movesAllowedList

Αυτή η μέθοδος καλείται πριν την σειρά του κάθε παίκτη ώστε να καταρτιστεί μια λίστα με τις επιτρεπτές κινήσεις που μπορεί να επιλέξει για να πραγματοποιήσει. Διαβάζοντας την τρέχουσα κατάσταση του πίνακα δημιουργεί όλες τις νέες κινήσεις για να τοποθετηθούν μέσα στην λίστα και ενημερώνει τις δικές τους επιμέρους λίστες κατευθύνσεων με τις κατευθύνσεις στις οποίες βρήκε πιόνια για παγίδευση.

2.4.8 Η μέθοδος play

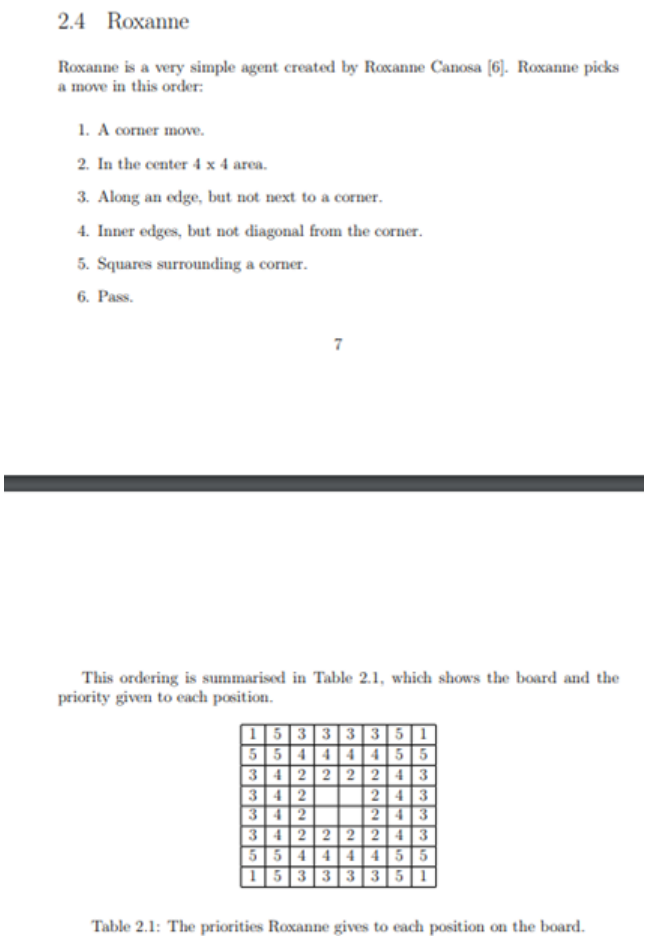
Είναι από τις βασικότερες μεθόδους του παιχνιδιού αφού πραγματοποιεί την κίνηση που επέλεξε ο παίκτης. Για την επιλεγμένη κίνηση, διαβάζει την λίστα directions της και εκτελεί την changeColor σε όσα Square χρειάζεται.

2.4.9 Η μέθοδος childBoard

Είναι μέθοδος κλειδί για την λειτουργία του Minimax αφού δημιουργεί έναν πίνακα παιχνιδιού, απόγονο του τρέχοντος, για κάθε κίνηση που έχει στην διάθεσή του ο CPU Player. Έτσι, έχοντας την λίστα με όλους τους πιθανούς μελλοντικούς πίνακες και με χρήση της evaluate επιλέγει τον βέλτιστο για την τρέχουσα κατάσταση.

2.4.10 Η μέθοδος evaluate

Είναι η μέθοδος που αποδίδει σε κάθε πίνακα (απόγονο του τρέχοντος) την αξία του για τον minimax. Ακολουθεί την παρακάτω λογική του agent Roxanne:



2.4 Roxanne

Roxanne is a very simple agent created by Roxanne Canosa [6]. Roxanne picks a move in this order:

1. A corner move.
2. In the center 4 x 4 area.
3. Along an edge, but not next to a corner.
4. Inner edges, but not diagonal from the corner.
5. Squares surrounding a corner.
6. Pass.

7

This ordering is summarised in Table 2.1, which shows the board and the priority given to each position.

1	5	3	3	3	3	5	1
5	5	4	4	4	4	5	5
3	4	2	2	2	2	4	3
3	4	2			2	4	3
3	4	2			2	4	3
3	4	2	2	2	2	4	3
5	5	4	4	4	4	5	5
1	5	3	3	3	3	5	1

Table 2.1: The priorities Roxanne gives to each position on the board.

Figure 1: Επεξήγηση Roxanne

Πρόκειται για απόσπασμα μελέτης φοιτητών ενός κολεγίου της Αυστραλίας, σχετικά με το παιχνίδι και το μαθηματικό και αλγοριθμικό υπόβαθρο του. Ουσιαστικά αποδίδουμε αξίες σε κάθε θέση εντός του πίνακα με βάση την αξία τους για την εξέλιξη του παιχνιδιού. Τέλος, υλοποιείται μια απλή ευρετική συνάρτηση που λαμβάνει υπόψιν την αξία των θέσεων που θα καταλαμβάνει ο κάθε παίκτης αλλά και τις κινήσεις που θα έχει διαθέσιμες στην επόμενη φάση του παιχνιδιού.

2.4.11 Η μέθοδος putStarOnPossibleMoves

Η μέθοδος αυτή συμβάλλει στην γραφική αναπαράσταση των δεδομένων του πίνακα του παιχνιδιού. Ορίζει προσωρινά ως $*$ το χρώμα κάθε κενού Square που μπορεί ο παίκτης να κινηθεί, εκτυπώνει τον προσωρινό ενημερωμένο πίνακα και στην συνέχεια τον επαναφέρει στην αρχική του κατάσταση επαναφέροντας ξανά το χρώμα των κελιών που είχε αλλάξει.

2.5 Η κλάση Player

Κλάση που αντιπροσωπεύει τους παίκτες του παιχνιδιού. Αν πρόκειται για άνθρωπο, αποθηκευέι απλώς το όνομά του και το χρώμα του, ενώ αν πρόκειται για CPU τότε αποθηκεύει και το μέγιστο βάθος αναζήτησης με βάση την δυσκολία που το ορίζει ο πελάτης, και υλοποιεί τον minimax αλγόριθμο.

2.5.1 Constructors - Getters - Setters

Ο κατασκευαστής καλείται στην main με στόχο την δημιουργία των δύο παικτών που θα συμμετάσχουν σε αυτή την παρτίδα. Αν πρόκειται για άνθρωπο θέτει το $maxDepth = 0$ ενώ αν πρόκειται για CPU τότε χρησιμοποιεί το depth που προκύπτει από την δυσκολία που θα επιλέξει ο παίκτης. Οι getters και οι setters αξιοποιούνται εντός της ροής του παιχνιδιού για να μπορεί μεταξύ άλλων να εκτυπώνει προσωποποιημένα μηνύματα και να αποκαλεί τους παίκτες με τα ονόματά τους.

2.5.2 Η μέθοδος isPlayingNow

Ερώτηση λογικής που εξετάζει αν το χρώμα - παίκτης που είναι η σειρά του να παίξει ταυτίζεται με το αντικείμενο - παίκτη

2.5.3 Οι μέθοδοι του Minimax

Οι μέθοδοι max και min αποτελούν κομμάτι της ροής του αλγορίθμου minimax, καλούνται αναδρομικά εντός του σώματός της μεθόδου Minimax και αναλύονται διεξοδικά στην συνέχεια.

2.6 Η κλάση Othello

Αποτελεί την κύρια κλάση του προγράμματος και ουσιαστικά το εκτελέσιμο αρχείο αυτού, καθώς περιλαμβάνει την μέθοδο main. Χαρακηριστικά της είναι ένας νέος πίνακας παιχνιδιού και μια μεταβλητή playingNow που υποδεικνύει ποιανού σειρά είναι να παίξει. Πρώτα παίζουν πάντα τα μαύρα οπότε αρχικοποείται σε αυτή τη τιμή. Η main αναλύεται διεξοδικά στην συνέχεια.

3 Ο αλγόριθμος Minimax

Δεδομένης μιας κατάστασης του παιχνιδιού, ο αλγόριθμος αναζήτησης μεγίστου - ελαχίστου (Minimax) καλείται να αποφασίσει ποια θα είναι η επόμενη κίνησή του έναντι του αντιπάλου. Χρησιμοποεί έναν απλό αναδρομικό αλγόριθμο αναζήτησης σε βάθος για τον υπολογσιμό κάθε διάδοχης κατάστασης υλοποιώντας τις μεθόδους childBoard, evaluate και minimax. Αν δεν υπάρχουν διαθέσιμες κινήσεις και συνεπώς ούτε δέντρο πινάκων, επιστρέφει μήνυμα πως ο CPU παίκτης δεν έχει διαθέσιμες κινήσεις και χάνει την σειρά του, τερματίζοντας τον αλγόριθμο. Αλλιώς, η Minimax καλεί την max εαν παίζει ο μαύρος παίκτης ή την min αν παίζει ο λευκός. Απο κει και έπειτα καλούνται εναλλάξ και αναδρομικά για να αποτιμηθεί η βέλτιστη κίνηση. Δημιουργεί ένα δέντρο από πίνακες για κάθε κίνηση που μπορεί να προκύψει από κάθε προηγούμενη κατάσταση, φτάνωντας σε βάθος μέχρι το maxDepth που έχει οριστεί. Ο max αρχικοποιείται σε μια πάρα πολύ μικρή αξία και ο min σε μια πάρα πολύ μεγάλη ώστε να εξασφαλιστεί η αποδοτικότητα του αλγορίθμου που επιδιώκει το ακριβώς αντίθετο ως αποτέλεσμα.

AB prunning

Εντός των δύο μεθόδων max και min υλοποείται η ιδέα του κλαδέματος άλφα - βήτα. Η συγκεκριμένη τεχνική εφαρμόζεται σε ένα συνηθισμένο δέντρο minimax και επιστρέφει το

ίδιο αποτέλεσμα με αυτόν, εξαλείφοντας ωστόσο στην πορεία κλάδους του δέντρου που δεν επηρεάζουν το αποτέλεσμα, όπου:

$\alpha =$ η τιμή της καλύτερης (μέγιστης) επιλογής μέχρι στιγμής σε οποιδήποτε σημείο κατά μήκος της διαδρομής για τον MAX

$\beta =$ η τιμή της καλύτερης (ελάχιστης) επιλογής μέχρι στιγμής σε οποιδήποτε σημείο κατά μήκος της διαδρομής για τον MIN.

Ο αλγόριθμος ενημερώνει τα α,β καθώς προχωρά και "κλαδεύει" τους κλάδους ενός κόμβου, τερματίζοντας την αναδρομή, όταν γίνει γνωστό ότι η τιμή του τρέχοντος κόμβου είναι χειρότερη από το α ή το β για τον max ή τον min αντίστοιχα.

4 Η main μέθοδος

Η κεντρική μέθοδος του προγράμματος εμπεριέχεται στην κλάση Othello. Η ροή του προγράμματος βρίσκεται εμφωλευμένη μέσα σε ένα do-while loop ώστε κάθε φορά που τελειώνει μια παρτίδα, ο χρήστης να έχει την επιλογή να ξαναπαίξει χωρίς να χρειαστεί να ανοίξει εκ νέου την εφαρμογή. Η αλληλεπίδραση με τον παίκτη ξεκινάει με ένα μήνυμα που ζητάει από τον παίκτη να αποφασίσει πώς θέλει να παίξει. Υπάρχουν τρείς επιλογές, 0, 1 ή 2 ανάλογα από το πλήθος των παικτών - ανθρώπων που θα συμμετάσχουν. Στην περίπτωση ενός ανθρώπινου παίκτη, το πρόγραμμα ζητάει και την δυσκολία που θέλει ο παίκτης για τον αντίπαλό του, αντιστοιχίζοντας της απάντηση του χρήστη σε κάποιο maxDepth για να δωθεί σαν όρισμα στον minimax.

4.1 Η μέθοδος Demo

Η μέθοδος αυτή αντιπροσωπεύει την λειτουργία παρουσίασης του προγράμματος για να κάνει μια επίδειξη του πως δουλέυει ο αλγόριθμος χωρίς την παρέμβαση του χρήστη. Ζητάει από τον χρήστη το χρόνο που επιθυμεί να σταματάει το παιχνίδι μετά απο κάθε σειρά ώστε να μπορεί να παρακολουθήσει το παιχνίδι που εξελίσσεται και στη συνέχεια δημιουργεί δύο παίκτες CPU με ένα default βάθος αναζήτησης ίσο με 3. Το παιχνίδι εξελίσσεται με διαδοχικές κλήσεις το minimax για κάθε παίκτη και στο τέλος εκτυπώνονται τα ανάλογα μηνύματα τερματισμού.

4.2 Η μέθοδος PvC

Η μέθοδος *παίκτης-εναντίον-υπολογιστή (PLAYERvsCOMPUTER)* αφού λάβει το όνομα του χρήστη και την επιθυμία του να παίξει πρώτος ή δεύτερος, δημιουργεί δύο αντικείμενα Player: ένα για τον ανθρώπινο παίκτη με maxDepth όρισμα ίσο με 0 και ένα για τον CPU παίκτη με maxDepth όρισμα ίσο με την τιμή που δώθηκε από τον χρήστη πριν από την κλήση της PvC. Το παιχνίδι εξελίσσεται με εντολές του ανθρώπινου παίκτη όταν είναι η σειρά του ή με κλήση του mini-max όταν είναι η σειρά του CPU. Όταν εξετάζεται η σειρά του ανθρώπινου παίκτη, αν διαπιστωθεί πως δεν υπάρχει διαθέσιμη κίνηση τότε εμφανίζει το ανάλογο μήνυμα και προχωράει στον επόμενο, διαθέτοντας και έλεγχο για απο-φυγή εισόδου σε ατέρμονα βρόγχο όταν κανείς από τους δύο παίκτες δεν έχει διαθέσιμη κίνηση να κάνει. Στο τέλος εκτυπώνει ανάλογα και προσωποποιημένα μηνύματα με την έκβαση του παιχνιδιού.

4.3 Η μέθοδος PvP

Η μέθοδος *παίκτης-εναντίον-παίκτη (PLAYERvsPLAYER)* λαμβάνει τα ονόματα των παικτών και αποδίδει αυτοματοποιημένα το μαύρο χρώμα στον πρώτο παίκτη και το λευκό στον δεύτερο. Αρχικοποιεί δύο αντικείμενα Player με τα ονόματα και τα χρώματά τους και maxDepth ίσο με 0, αφου σε αυτήν τη λειτουργία του προγράμματος ο minimax δεν αξιοποιείται κάπως. Το παιχνίδι εξελίσσεται με αλληλεπίδραση μεταξύ παικτών και συστήματος ενώ πραγματοποιείται έλγχος αποφυγής ατέρμονος βρόγχου και εκτυπώνονται τα σχετικά προσωποποιημένα μηνύματα στο τέλος του.

5 Το παιχνίδι

Λίγα λόγια για το παιχνίδι. Παίζεται σε ένα τετράγωνο ταμπλό με 8×8 τετράγωνες θέσεις οπού τοποθετούνται στρογγυλά πιόνια δύο χρωμάτων. Το παιχνίδι ξεκινάει με κάθε παίκτη να βάζει δύο πιόνια στο κέντρο του πίνακα σε διαγώνια διάταξη, όπως φαίνεται στην εικόνα

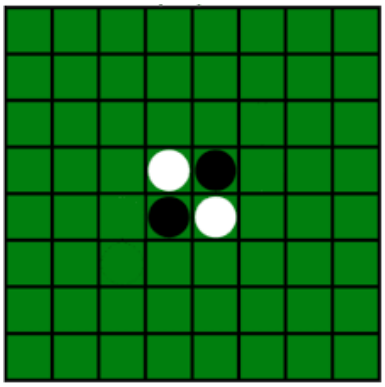


Figure 2: Αρχική Κατάσταση παιχνιδιού

Οι παίκτες τοποθετούν διαδοχικά πιόνια στον πίνακα με στόχο να παγιδεύσουν αντίπαλα πιόνια μεταξύ δύο δικών τους πιονιών. Αυτό αποτελεί περιορισμό στο παιχνίδι καθώς για να τοποθετήσει ένας παίκτης πιόνι σε κάποια συγκεκριμένη θέση θα πρέπει αυτή η κίνησή τυο να επιφέρει παγίδεση ενός τουλάχιστον αντίπαλου πιονιού. Τα παγιδευμένα πιόνια αλλάζουν χρώμα, παίρνωντας το χρώμα του παίκτη που τα παγίδευσε. Αν ένας παίκτης δεν έχει διαθέσιμη επιτρεπτή κίνηση να πραγματοποιήσει και παίζει ο επόμενος. Αν και ο επόμενος παίκτης δεν έχει διαθέσιμη επιτρεπτή κίνηση να πραγματοποιήσει τότε το παιχνίδι τελειώνει και προσμετράται το σύνολο των μαύρων και των λευκών πιονιών. Διαφορετικά το παιχνίδι συνεχίζεται μέχρι να γεμίσει το ταμπλό του παιχνιδιού, οπότε και προσμετράται το σύνολο των μαύρων και των λευκών πιονιών. Νικητής είναι ο παίκτης που μετά την τελική καταμέτρηση έχει περισσότερα πιόνια του χρώματός του στο ταμπλό από ότι ο αντίπαλος. Η περίπτωση της ισοπαλλίας είναι πιθανή και συνεπώς επιτρεπτή.

6 Επικοινωνία

ΓΙΑΝΝΗΣ - giannis.papachristou@yahoo.gr
ΦΩΤΗΣ - fotpanos@gmail.com