

Σοφία Ζαφείρη (3220288)  
Νίκος Μαυράκης (3230328)  
Δημήτριος Καρλής (3220075)

# 1η Εργασία - Reversi

## 1. Τρόπος Χρήσης Προγράμματος

Η κλάση GameWindow χρησιμοποιήθηκε για την αλληλεπίδραση με τον χρήστη μέσα από το terminal.

Επιλογή βάθους Minimax

Ο χρήστης εισάγει το βάθος αναζήτησης (1-10)

Το πρόγραμμα ελέγχει:

- Αν η τιμή είναι ακέραιος

- Αν βρίσκεται μέσα στο επιτρεπτό εύρος

- Αν όχι, εμφανίζει μήνυμα σφάλματος και ζητά ξανά είσοδο

Επιλογή αν ο χρήστης παίζει πρώτος

Ο χρήστης εισάγει γ ή n:

Το πρόγραμμα δέχεται επίσης ελληνικούς χαρακτήρες

Κλήση Game Window με παραμέτρους

```
new GameWindow(maxDepth, userPlaysFirst);
```

Δημιουργία παικτών με βάση την επιλογή του χρήστη

Αν ο χρήστης παίζει πρώτος(μαύρα):

```
player1 = new HumanPlayer(Board.B);
```

```
player2 = new MinimaxPlayer(Board.W, maxDepth);
```

Αν ο χρήστης παίζει δεύτερος(άσπρα):

```
player1 = new MinimaxPlayer(Board.B, maxDepth);
```

```
player2 = new HumanPlayer(Board.W);
```

## 2. Αρχιτεκτονική

Χρησιμοποιήσαμε έτοιμο ελεύθερα διαθέσιμο λογισμικό για τη διεπαφή χρήσης, την παράσταση των καταστάσεων, την εύρεση των επιτρεπτών κινήσεων από μία κατάσταση (<https://github.com/arminkz/Reversi>).

Ο κώδικας αποτελείται από τους φακέλους `src/game`, όπου βρίσκονται διάφορες βοηθητικές κλάσεις όπως η `BoardHelper` και άλλες που βοηθούν στην διεπαφή του χρήστη. Χρησιμοποιούμε την `BoardHelper` στις εύρεση συγκεκριμένων χαρακτηριστικών χρήσιμων για τις ευρετικές και στον αλγόριθμο `minimax` για την εύρεση χρήσιμων δεδομένων, όπως πχ. την παραγωγή όλων των παιδιών-καταστάσεων μιας κατάστασης. Η `GamePanel` παρέχει την υλοποίηση για τη διεξαγωγή του παιχνιδιού, την αλλαγή σειράς παίκτη, τις κατάλληλες ενημερώσεις στη διεπαφή του χρήστη, κτλ.

Το φάκελο `player/ai` όπου αναπτύξαμε τον κώδικα για τη `minimax`. Για την αναπαράσταση καταστάσεων χρησιμοποιούμε τη κλάση `Board`, η οποία χρησιμοποιεί στοιχεία της `BoardHelper` για την υλοποίησή της, καθώς και τη κλάση `Move` για την αναπαράσταση των κινήσεων που γίνονται στις καταστάσεις. Επιπλέον έχουμε τη κλάση `HumanPlayer`, η οποία χρησιμοποιείται για τον ανθρώπινο παίκτη.

Τέλος, έχουμε το φάκελο `player/evaluator` όπου βρίσκεται ο κώδικας για τις ευρετικές.

## 3. Μέθοδοι Τεχνητής Νοημοσύνης

Ο αλγόριθμος που υλοποιήσαμε ώστε να παίζει ενάντια ενός ανθρώπινου παίκτη είναι ο `Minimax` με πριόνισμα  $\alpha$ - $\beta$ .

Ο αλγόριθμος βρίσκεται στο αρχείο `src/player/ai/MinimaxPlayer.java` και αποτελεί υποκλάση του `GamePlayer`. Έχουμε κατασκευαστή

```
public MinimaxPlayer(int mark, int maxDepth)
```

ο οποίος δέχεται ως ορίσματα τι πούλι χρησιμοποιεί ο υπολογιστής (`mark`), καθώς και το μέγιστο βάθος αναζήτησης (`maxDepth`).

Σε κάθε κίνηση που είναι να κάνει ο υπολογιστής, καλείται η συνάρτηση

```
public Move play(Board board)
```

η οποία δέχεται ως όρισμα την εκάστοτε θέση του πίνακα που βρίσκεται το παιχνίδι εκείνη τη στιγμή.

Στη συνέχεια καλείται η συνάρτηση

```
public Move MiniMax(Board board)
```

η οποία τρέχει το maximizing, καθώς γνωρίζουμε ότι το play καλείται μόνο όταν είναι σειρά του υπολογιστή να παίξει και ο υπολογιστής στη σειρά του προσπαθεί να κάνει maximizing στον αλγόριθμο minimax

Η συνάρτηση max είναι η

```
Move max(Board board, int depth, int alpha, int beta)
```

η οποία αρχικά ελέγχει αν βρισκόμαστε σε τελική κατάσταση και αν όχι, βρίσκουμε όλες τις πιθανές θεσεις-παιδιά της τωρινής θέσης και τρέχουμε το min για αυτά προσπαθώντας να βρούμε την μέγιστη τιμή, δηλαδή την κατάσταση την οποία συμφέρει περισσότερο τον υπολογιστή ώστε να παίξουμε την κίνηση που οδηγεί σε αυτή τη κατάσταση. Μόλις βρεθούμε σε τελική κατάσταση ή στο μέγιστο επιτρεπτό βάθος, επιστρέφουμε την κίνηση με εκτίμηση τιμής που βρίσκεται από την ευρετική που χρησιμοποιούμε. Επιπλέον, κάνουμε χρήση α-β πριονίσματος.

Αντίστοιχα, έχουμε τη συνάρτηση min

```
Move min(Board board, int depth, int alpha, int beta)
```

η οποία λειτουργεί όπως η max με τη διαφορά ότι προσπαθεί να βρει τη χαμηλότερη τιμή, δηλαδή την βέλτιστη κίνηση για τον αντίπαλο, στη συγκεκριμένη περίπτωση τον άνθρωπο με τον οποίο παίζει εναντίον ή κάποιο άλλο AI.

## 4. Η Ευρετική Συνάρτηση

Η ευφυΐα του παίκτη στο Othello/Reversi βασίζεται στην **Ευρετική Συνάρτηση Αξιολόγησης (H)**, η οποία εκτιμά τις καταστάσεις που δημιουργούνται κατά τη διάρκεια του παιχνιδιού. Η υλοποίηση της βασίζεται στην κλάση **StaticEvaluator**, η οποία εφαρμόζει διάφορες ευρετικές με δυναμικά βάρη που προσαρμόζονται ανάλογα με τη φάση στην οποία βρίσκεται το παιχνίδι.

### 4.1 Οι Επιμέρους Ευρετικές Συνιστώσες

Η κάθε συνιστώσα υπολογίζει τη διαφορά μεταξύ του παίκτη Player και του αντιπάλου Opponent, στο εύρος [-100,100] εξασφαλίζοντας συγκρισιμότητα:

#### 4.1.1 Διαφορά Δίσκων (evalDiscDiff)

Η λειτουργία της **evalDiscDiff** είναι ο υπολογισμός της διαφοράς στον αριθμό των δίσκων Player – Opponent, καθώς είναι η βασική συνιστώσα για το τέλος του παιχνιδιού (End Game), όπου η νίκη καθορίζεται από τον μέγιστο αριθμό δίσκων.

#### 4.1.2 Κινητικότητα (evalMobility)

Η λειτουργία της **evalMobility** είναι ο υπολογισμός της διαφοράς στον αριθμό των διαθέσιμων έγκυρων κινήσεων μεταξύ του Player και του Opponent. Η υψηλή κινητικότητα παρέχει ευελιξία, ελαχιστοποιεί τις αναγκαστικές κινήσεις και αυξάνει την πιθανότητα απόκτησης στρατηγικών θέσεων. Για αυτό το λόγο είναι η πιο σημαντική συνιστώσα για το αρχικό στάδιο του παιχνιδιού.

#### 4.1.3 Κατοχή Γωνιών (evalCorner)

Η λειτουργία της **evalCorner** είναι η καταγραφή της διαφοράς στον αριθμό των κατειλημμένων γωνιών του ταμπλό. Οι γωνίες βρίσκονται σε αμετάβλητες θέσεις και η κατοχή τους είναι στρατηγικής σημασίας σε όλες τις φάσεις του παιχνιδιού και έχει το υψηλότερο στατικό βάρος.

#### 4.1.4 Θέση Δίσκων (evalBoardMap)

Η λειτουργία της **evalBoardMap** είναι ο υπολογισμός ενός συνολικού σκορ θέσης με βάση έναν πίνακα βαρών θέσης W. Οι γωνίες του πίνακα W έχουν το μέγιστο θετικό βάρος (π.χ. 200), ενώ οι θέσεις δίπλα τους X-squares και C-squares έχουν αρνητικό βάρος (π.χ. -100, -200) για τις αποφεύγει ο αλγόριθμος, καθώς θεωρούνται 'επικίνδυνες' θέσεις.

Αυτή η ευρετική χαρακτηρίζεται από τη δυναμική προσαρμογή της στη μηδένιση των βαρών του πίνακα W στα γειτονικά τετράγωνα μιας κατακτημένης γωνίας. Αυτό γίνεται επειδή, μόλις κατακτηθεί μια γωνία, η κατοχή των γύρω θέσεων (που ήταν αρχικά επικίνδυνες) καθίσταται ασφαλής και το αρνητικό τους βάρος δεν είναι πλέον απαραίτητο.

#### 4.1.5 Σύνορο Δίσκων(evalFrontierNew)

Η λειτουργία της **evalFrontierNew** είναι η μέτρηση της διαφοράς στον αριθμό των δίσκων συνόρου οι οποίοι βρίσκονται δίπλα σε τουλάχιστον ένα κενό τετράγωνο. Οι δίσκοι συνόρου κινδυνεύουν περισσότερο να αναστραφούν από τον αντίπαλο, για αυτό το λόγο επιδιώκουμε την ελαχιστοποίηση τους.

#### 4.1.6 Σταθερότητα(evalStabilityNew)

Η λειτουργία της **evalStabilityNew** είναι η μέτρηση της σταθερότητας των δίσκων, εστιάζοντας στις αλυσίδες δίσκων που ξεκινούν από τις γωνίες και εκτείνονται στις πλευρές του ταμπλό. Με αυτό το τρόπο παρέχεται μια

εκτίμηση των πραγματικά σταθερών δίσκων, οι οποίοι είναι προστατευμένοι και δεν μπορούν να αναστραφούν.

#### 4.1.7 Ισοτιμία(evalParity)

Η λειτουργία της **evalParity** είναι ο υπολογισμός της ισοτιμίας του αριθμού των υπολειπόμενων κενών τετραγώνων. Εάν ο αριθμός των κενών τετραγώνων είναι μονός, η τελευταία κίνηση ανήκει στον παίχτη που εκτελεί την τρέχουσα κίνηση σε σχέση με το MinMax, δίνοντας του ένα μικρό πλεονέκτημα.

## 4.2 Δυναμικά Βάρη

Τα δυναμικά βάρη χρησιμοποιούνται για την μέγιστη αποτελεσματικότητα του ευφυούς παίχτη, καθώς προσαρμόζονται κατάλληλα σε κάθε ευρετική συνιστώσα, ανάλογα με τη στρατηγική φάση του παιχνιδιού. Η αλλαγή φάσης καθορίζεται από τον συνολικό αριθμό των δίσκων **totalDiscs** που έχουν τοποθετηθεί στο ταμπλό. Αυτή η προσέγγιση εξασφαλίζει ότι η τεχνητή νοημοσύνη υιοθετεί την κατάλληλη στρατηγική εστίαση σε κάθε στάδιο.

Φάση	Δίσκοι <b>totalDiscs</b>	Στρατηγική παιχνιδιού
<b>Early Game</b>	<b>&lt;=20</b>	Επιθετική κατοχή θέσεων με έντονη κινητικότητα σε θέση και γωνίες
<b>Mid Game</b>	<b>21 - 52</b>	Μείωση της κινητικότητας και αύξηση της διαφοράς των δίσκων
<b>End Game</b>	<b>&gt;= 53</b>	Κυριαρχία στη διαφορά δίσκων και στην ισοτιμία

Συμπερασματικά, με τα δυναμικά βάρη η ευρετική συνάρτηση προσαρμόζεται πιο ορθά σε κάθε φάση του παιχνιδιού, με αποτέλεσμα τη μέγιστη αποτελεσματικότητα του αλγορίθμου **MiniMax**.

## 5. Παραδείγματα χρήσης

Παιχνίδι ενάντια σε άνθρωπο. Ο άνθρωπος είναι με λευκά και το AI με μαύρα. Βρίσκεται στο αρχείο `play_example.mp4` ένα βίντεο παράδειγμα.