TrailGame: Αναλυτική Περιγραφή και Ανάλυση

Εισαγωγή

Το **TrailGame** αποτελεί μία εφαρμογή που αναπτύχθηκε με σκοπό την ενσωμάτωση τεχνητής νοημοσύνης (AI) σε ένα διαδραστικό περιβάλλον παιχνιδιού. Το παιχνίδι χρησιμοποιεί την τεχνική Minimax με περικοπή alpha-beta για τη λήψη αποφάσεων και εφαρμόζει ευρετικές για την αξιολόγηση των κινήσεων στον πίνακα παιχνιδιού. Οι παίκτες έχουν τη δυνατότητα να διαγωνιστούν εναντίον του AI, επιλέγοντας στρατηγικές κινήσεις που βασίζονται στη θέση και τη συνδεσιμότητα.

Περιγραφή του Παιχνιδιού

Στόχος Παιχνιδιού: Οι παίκτες ανταγωνίζονται για να κυριαρχήσουν στον πίνακα παιχνιδιού, ο οποίος είναι ένας δισδιάστατος πίνακας προκαθορισμένου μεγέθους(5X5). Κάθε παίκτης αντιπροσωπεύεται από το χρώμα μπλε και ο υπολογιστής από το χρώμα κόκκινο. Στόχος του παιχνιδιού είναι να δημιουργηθεί η μεγαλύτερη δυνατή ακολουθία από χρωματισμένα κουτάκια ώστε να κερδίσουν οι παίκτες τον υπολογιστή. Αυτό μπορούν να το κάνουν με έξυπνες στρατηγικές κινήσεις, μεγαλώνοντας την δική τους ακολουθία, εμποδίζοντας ταυτόχρονα την ανάπτυξη ακολουθίας του υπολογιστή ώστε να καταλήξουν στην νίκη. Ο παίκτης παίζει πρώτος και έχει 13 κινήσεις, ενώ ο υπολογιστής παίζει 12 φορές.

Κανόνες Παιχνιδιού

1. Ο παίκτης έχει το χρώμα μπλε

2. Οι κινήσεις είναι έγκυρες μόνο όταν επιλέγεται ένα κενό κελί.

3. Η AI αξιολογεί όλες τις πιθανές κινήσεις και λαμβάνει αποφάσεις χρησιμοποιώντας τον αλγόριθμο Minimax.

4. Το παιχνίδι ολοκληρώνεται όταν δεν υπάρχουν διαθέσιμες κινήσεις στον πίνακα, εμφανίζοντας μήνυμα στην οθόνη σχετικά με το ποιος είναι ο νικητής.

Τεχνική Υλοποίηση

Το παιχνίδι αναπτύχθηκε με τη γλώσσα προγραμματισμού **Java** και χρησιμοποιεί διάφορες κλάσεις για την οργάνωση της λειτουργικότητάς του. Παρακάτω παρουσιάζεται η ανάλυση κάθε κλάσης.

1. Main

Η κλάση **Main** είναι το σημείο εκκίνησης του παιχνιδιού. Δημιουργεί το γραφικό περιβάλλον του παιχνιδιού χρησιμοποιώντας τη βιβλιοθήκη **Swing** και αρχικοποιεί τα βασικά στοιχεία:

- Καθορίζει τα χρώματα του AI και του παίκτη.

- Δημιουργεί το παράθυρο παιχνιδιού (JFrame) και το **BoardPanel**.

2. Board

Η κλάση **Board** αντιπροσωπεύει τον πίνακα παιχνιδιού και περιλαμβάνει:

- **Σταθερές χρώματος** (`RED`, `BLUE`, `EMPTY`) για την αναπαράσταση των κελιών.

- **Λειτουργίες όπως**:

  - **makeMove**: Καταγράφει μια νέα κίνηση στον πίνακα.

  - **isValidMove**: Ελέγχει αν μια κίνηση είναι έγκυρη.

  - **evaluate**: Υπολογίζει τη "βαθμολογία" του πίνακα για την AI.

  - **heuristics**: Υλοποιεί ευρετικές που λαμβάνουν υπόψη τη συνδεσιμότητα, την απόσταση από το κέντρο και τα ελεύθερα μπλοκ γύρω από κάθε στοιχείο.

- Υποστήριξη για τη δημιουργία παιδιών πινάκων μέσω της μεθόδου **getChildren**.

3. Move

Η κλάση **Move** αποθηκεύει τις πληροφορίες μιας κίνησης:

- Συντεταγμένες (row, col).

- Την τιμή της κίνησης (χρησιμοποιείται για αξιολόγηση από την AI).

- Περιλαμβάνει μεθόδους **get** και **set** για ευκολότερη πρόσβαση στα δεδομένα.

4. Player

Η κλάση **Player** αντιπροσωπεύει τον AI παίκτη και υλοποιεί τον αλγόριθμο Minimax με περικοπή alpha-beta:

- **MiniMax**: Καλεί τις λειτουργίες **max** και **min** για την ανάλυση των δυνατών κινήσεων.

- **max** και **min**: Υπολογίζουν την καλύτερη κίνηση για τον AI ή τον αντίπαλο, αντίστοιχα, με βάση την ευρετική αξιολόγηση.

5. Component

Η κλάση **Component** χρησιμοποιείται για την αποθήκευση και διαχείριση δεδομένων σχετικά με τα συνδεδεμένα στοιχεία στον πίνακα.

6. Point

Η κλάση **Point** αναπαριστά μια θέση στον πίνακα και υποστηρίζει λειτουργίες σύγκρισης (**equals**) και δημιουργίας μοναδικών κλειδιών (**hashCode**).

7.BoardPanel

Η κλάση **BoardPanel** κληρονομεί από την JPanel και παρέχει ένα γραφικό περιβάλλον για το παιχνίδι. Δημιουργεί έναν πίνακα κουμπιών (buttons) που αναπαριστούν το ταμπλό του παιχνιδιού και διαχειρίζεται τις κινήσεις του χρήστη και της AI μέσω διαδραστικών στοιχείων.

Λειτουργίες:

1. Το ταμπλό αρχικοποιείται με λευκά κουμπιά (Color.WHITE).
2. Ο χρήστης κάνει την κίνηση του πατώντας ένα κουμπί. Αν η κίνηση είναι έγκυρη:
    - Το χρώμα του κουμπιού αλλάζει σύμφωνα με τον παίκτη.
    - Ελέγχεται αν το παιχνίδι έχει τελειώσει.
3. Αν το παιχνίδι συνεχίζεται, η AI κάνει την κίνησή της χρησιμοποιώντας **MiniMax**.
4. Η διαδικασία επαναλαμβάνεται έως ότου το παιχνίδι τερματιστεί.
5. Η μέθοδος **evaluateGame** αποφασίζει τον νικητή και εμφανίζει το αντίστοιχο μήνυμα.

Λειτουργία του AI

Η τεχνητή νοημοσύνη στο **TrailGame** βασίζεται σε τρία κύρια στοιχεία:

1. Minimax με περικοπή alpha-beta

Ο αλγόριθμος **Minimax** χρησιμοποιείται για να αξιολογήσει όλες τις πιθανές κινήσεις του AI και του αντιπάλου. Η περικοπή **alpha-beta** μειώνει τον αριθμό των κόμβων που εξετάζονται, βελτιώνοντας την απόδοση.

2. Ευρετικές

Η AI χρησιμοποιεί **ευρετικές** για την αξιολόγηση του πίνακα:

- **H1**: Υπολογίζει τη διαφορά μεταξύ του μεγαλύτερου συνδεδεμένου συνόλου κελιών του AI και του αντιπάλου.

- **H2**: Βασίζεται στην απόσταση μεταξύ των συνδεδεμένων συνόλων του AI.

- **H3**: Αξιολογεί την απόσταση των κελιών του AI από το κέντρο του πίνακα.

- **H4**: Εξετάζει τα ελεύθερα μπλοκ γύρω από κάθε συνδεδεμένο σύνολο.

3.Γραφικό Περιβάλλον

Το παιχνίδι χρησιμοποιεί τη βιβλιοθήκη **Swing** για τη δημιουργία του γραφικού περιβάλλοντος. Το παράθυρο παιχνιδιού (**JFrame**) περιλαμβάνει το **BoardPanel**, όπου εμφανίζεται ο πίνακας και οι κινήσεις των παικτών. Ο σχεδιασμός είναι απλός και επικεντρώνεται στη λειτουργικότητα.

Προκλήσεις υλοποίησης

Κατά την ανάπτυξη της εφαρμογής αντιμετωπίσαμε αρκετές προκλήσεις. Αρχικά, επειδή είναι ένα παιχνίδι που σκεφτήκαμε και υλοποιήσαμε εμείς, έπρεπε να καθορίσουμε την λειτουργία και τους κανόνες από την αρχή, ώστε να το προσεγγίσουμε σωστά και ταυτόχρονα να είναι διασκεδαστικό αλλά και να ανταποκρίνεται στις απαιτήσεις της εργασίας. Ύστερα, πρόκληση αποτέλεσε και σχεδιασμός αποτελεσματικών ευρετικών για την αξιολόγηση του πίνακα παιχνιδιού, καθώς απαιτεί βαθιά κατανόηση της στρατηγικής. Μια κακή ευρετική μπορεί να οδηγήσει σε μη ιδανικές αποφάσεις από την AI. Τέλος, ο καθορισμός του αριθμού και της σημαντικότητας των ευρετικών, ήταν επίσης μία δυσκολία που αντιμετωπίσαμε, καθώς με μία μικρή αλλαγή στα βάρη, άλλαζε όλη την λειτουργικότητα. Θέλαμε να έχει αρκετές ευρετικές ώστε να παίρνει έξυπνες και πολύπλοκες αποφάσεις, όχι όμως τόσες πολλές ώστε οι αποφάσεις που έπαιρνε αν είναι τόσο πολύπλοκες ώστε να μην καταλαβαίνουμε την λειτουργία του κώδικα.

Συμπεράσματα

Το **TrailGame** είναι ένα εξαιρετικό παράδειγμα ενσωμάτωσης τεχνητής νοημοσύνης σε ένα παιχνίδι στρατηγικής. Μέσα από την ανάπτυξή του:

- Κατανοήθηκε η εφαρμογή του αλγορίθμου **Minimax** ως προ την λήψη αποφάσεων με περικοπή **alpha-beta,** ως προς την βελτίωση της απόδοσης του αριθμού των υπολογισμών.

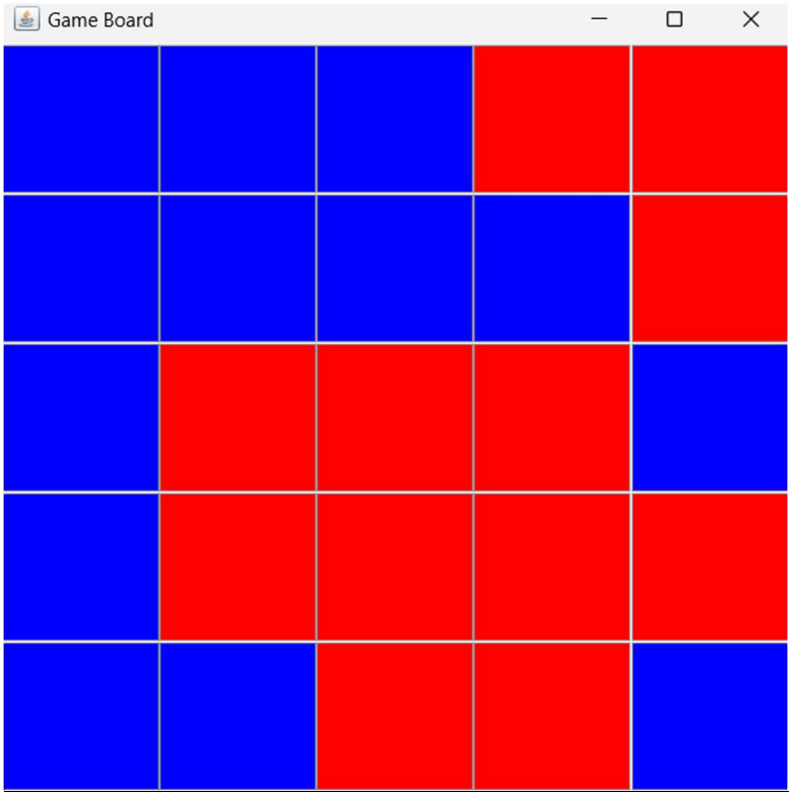
- Εξετάστηκε η χρήση πολλών **ευρετικών** για τη λήψη αποφάσεων με αντίστοιχα βάρη για την καλύτερη λήψη αποφάσεων.

- Αναπτύχθηκαν δεξιότητες προγραμματισμού με τη χρήση γραφικών διεπαφών (**Swing**) και δομών δεδομένων.

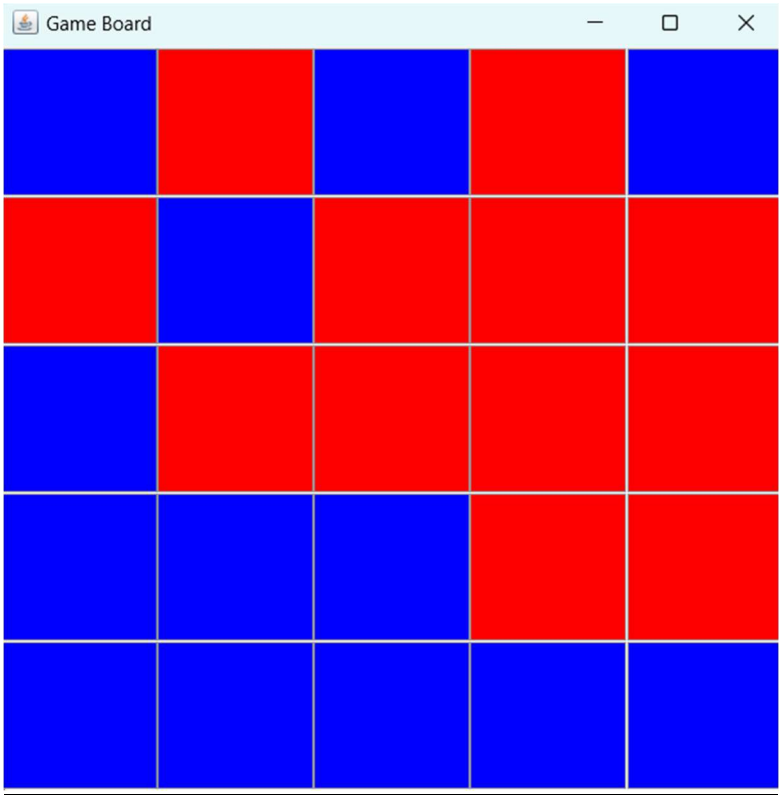
Καλές στρατηγικές για τον παίκτη

Γενικά με την συνεχή προσομοίωση του παιχνιδιού καταλήξαμε στο ότι η καλύτερη στρατηγική που μπορεί να ακολουθήσει ο παίκτης είναι να ξεκινάει γύρω από το κέντρο του πίνακα και όχι από τα άκρα. Έτσι μπορεί και να αναπτύξει πιο εύκολα μεγάλη ακολουθία και μπορεί και να εμποδίσει την ανάπτυξη της ακολουθίας του αντιπάλου.

Παράδειγμα μίας Παρτίδας όπου νικάει ο παίκτης



Παράδειγμα μίας Παρτίδας όπου χάνει ο παίκτης



Πηγές

-Για το Java Swing:

Βασιστήκαμε στο documentation και σε βίντεο στο youtube:
https://docs.oracle.com/javase/tutorial/uiswing/]

[https://www.youtube.com/watch?v=Kmgo00avvEw&t=14275s]s

- **Αλγόριθμος Minimax**:

Βασιστήκαμε κυρίως στις διαφάνειες του μαθήματος, σε αντίστοιχα βίντεο στο youtube και σε site στο internet όπως geeksforgeeks:

[https://www.appliedaicourse.com/blog/min-max-algorithm-in-artificial-intelligence/]

[https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/]