הצעת פרויקט – י"ג הנדסת תוכנה

שם הפרויקט: שחמט קלאסי

שם הסטודנט: עידן חרמש

תעודת זהות: 216094680

שם מכללה: מכללת אורט הרמלין נתניה

סמל המוסד: 471029



אתגרים תכנוניים בפיתוח שחקן ממוחשב במשחקי אסטרטגיה

האתגר בפיתוח שחקן ממוחשב לשחמט הוא ההתמודדות עם המורכבות הרבה הנובעת ממספר האפשרויות העצום בכל תור, שמוביל למספר בלתי נתפס של מצבים אפשריים לאורך המשחק. במשחקים פשוטים יותר, ניתן להשתמש בשיטות חיפוש סטנדרטיות, אך שחמט דורש גישה מתקדמת יותר, המשלבת לא רק חישוב טכני של מהלכים אלא גם הבנה אסטרטגית של המשחק.

כדי להתמודד עם האתגר, אפתח אלגוריתם שמסוגל לנתח באופן חכם את המצב הנוכחי על הלוח, לבחור את המהלך האופטימלי ולהתאים אותו לסיטואציות משתנות. האלגוריתם צריך לשלב איזון בין פעולות התקפיות והגנתיות, לקחת בחשבון עקרונות כמו שליטה במרכז הלוח, פיתוח כלים ושימור יתרון יחסי, וגם להתמודד עם רמות שונות של אי-ודאות שנובעות מתגובות היריב. יתר על כן, השחקן הממוחשב צריך להיות גמיש מספיק כדי להתאים את סגנון המשחק שלו לסגנון של יריבים שונים, בין אם מדובר באסטרטגיה שמרנית או אגרסיבית יותר.

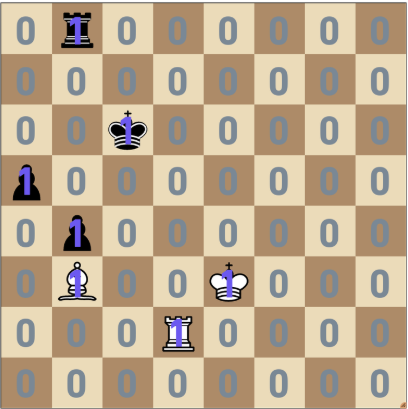
גישה זו מבטיחה שהשחקן הממוחשב לא רק יבצע מהלכים נכונים מבחינה חישובית, אלא גם יפגין הבנה עמוקה יותר של עקרונות המשחק, מה שהופך אותו ליריב ראוי גם לשחקנים אנושיים מנוסים..

**מבנה נתונים שהשתמש בהן בפרוייקט :**

**מבני הנתונים בהם נעשה שימוש**

**Bitboard (long)1.**

מבנה הנתונים משמש לייצוג יעיל של לוח השחמט באמצעות מספרים ביטיים תשלים. כל ביט מייצג משבצת בלוח כאשר הביט דולק מתאר חתיכה מסויימת על הלוח ואם מכובה אומר שיש מקום ריק, כך שניתן להשתמש בפעולות ביטיות כדי לבצע חישובים מהירים במיוחד, כמו חישוב מהלכים חוקיים או בדיקות איומים. מבנה זה משמש בעיקר לשיפור הביצועים ולהפחתת צריכת הזיכרון, שכן הוא חוסך את הצורך בשימוש במערכים ומאפשר ביצוע חישובים באופן אופטימלי.



**HashMap (Map<Integer, Piece>).2**

מבנה הנתונים מפה משמש לשמירת מיקומי הכלים על הלוח. הוא ממפה כל אינדקס 0-63 (בהתאם למשבצת בלוח) לאובייקט המייצג כלי שחמט מסוים. היתרון המרכזי של מפה הוא גישה מהירה לכלי לפי המיקום שלו, ללא צורך בסריקה של כל הלוח. בנוסף, הוא מאפשר הוספה והסרה דינאמית של כלים, מה שמקל על עדכון מצב הלוח בזמן המשחק.

**3.תור קדימויות**

מבנה הנתונים תור קדימויות משמש למיון מהלכים לפי הציון שלהם באלגוריתם שבניתי. כל מהלך מקבל ציון המבוסס על שיקולים כמו לקיחת כלי יריב, שליטה במרכז הלוח, הגנה על כלים חשובים וכדומה. תור עדיפויות זה מסייע בבחירה המהירה של המהלך הטוב ביותר, שכן הוא תמיד ממיין את המהלכים כך שהמהלך עם הציון הגבוה ביותר ייבחר לביצוע.

**4. רשימה מקושרת דו כיוונית**

מבנה הנתונים תשלים משמש לשמירת היסטוריית המהלכים של המשחק, ומאפשר לבצע תשלים בקלות. כל מהלך נוסף לסוף הרשימה, וכאשר רוצים לחזור אחורה, פשוט מסירים את המהלך האחרון. שימוש בתשלים מתאים במיוחד למקרה זה, שכן הוספה והסרה של אלמנטים מתבצעות ביעילות גבוהה, ללא הצורך בהעתקת .זיכרון

**5. עץ**

מבנה הנתונים עץ משמש לתכנון מהלכים לעומק של כמה תורות קדימה. כל צומת בעץ מייצגת מצב לוח מסוים, והצמתים הבנים שלו מייצגים את המהלכים האפשריים מהמצב הזה. מבנה זה חיוני לניתוח עתידי של מהלכים ולשיפור רמת המשחק של הבוט , שכן הוא מאפשר לבדוק את ההשלכות של מהלכים שונים לפני קבלת החלטה.

**HashMap (Map<String, List<Move>>)6.**

מבנה הנתונים זה משמש לשמירת מהלכי פתיחות בשחמט עבור הבוט. כל מפתח מייצג רצף מהלכים מסוים מתחילת המשחק, והערך הוא רשימת מהלכים אפשריים להמשך. הדבר מאפשר לבוט לבצע פתיחות מוכרות ויעילות, כך שהמשחק יתנהל בצורה אסטרטגית בשלבים הראשונים.

**Top-Down Diagram**

הסבר על המערכת :

**:View**

התצוגה אחראית על הצגת מצב המשחק לשחקנים ומתן אפשרות לאינטראקציה עם הלוח. היא אינה מנהלת חוקים או מבצעת חישובים אלא רק מציגה את הנתונים כפי שהמודל קובע.

**מרכיבי התצוגה:**

**הצגת הודעות:**

תפקידו להציג לשחקנים הודעות רלוונטיות על מצב המשחק. זה כולל הודעות כמו "שח", "שח-מט", "תיקו", או התראות על מהלכים לא חוקיים. מרכיב זה עוזר למשתמש להבין את מצב המשחק ולקבל פידבק על פעולותיו.

**הצגת מצב המשחק:**

תצוגת מצב הלוח כוללת את מיקום הכלים בזמן אמת ואת היסטוריית המהלכים. זה החלק שאחראי לוודא שהשחקנים רואים תמונת מצב עדכנית של המשחק.

**הצגת מהלכים אפשריים:**

כאשר משתמש בוחר כלי, המערכת מסמנת לו את כל המשבצות האפשריות אליהן הוא יכול לזוז. סימון זה יכול להיות באמצעות צבעים, סימנים גרפיים, או אנימציות כדי לשפר את חוויית המשתמש.

**הזזת הכלים:**

מקבל קלט מהמשתמש (לדוגמה, קליק על כלי ולאחר מכן קליק על משבצת) ומעביר את הפקודה לבקר. אם המהלך חוקי, התצוגה מתעדכנת בהתאם ומזיזה את הכלי.

**תפקיד model**

המודל אחראי על ניהול חוקי המשחק, חישוב מהלכים, ושמירת המצב הנוכחי של הלוח. זהו החלק של המערכת שמכיל את ה"אינטליגנציה" של המשחק – הוא מחליט האם מהלך חוקי, מחשב מצבי לוח ומנהל מבני נתונים.

**הערכת עמדה:** פונקציה שמחשבת עד כמה המצב הנוכחי של הלוח טוב או רע לכל שחקן. פונקציה זו קריטית במערכות עם בוט,שבהן האלגוריתם צריך להחליט איזה מהלך לבצע.

**קביעת מצב המשחק:** זהו מרכיב שמנתח את מצב המשחק וקובע האם המשחק ממשיך או שהגיע לסיומו. כאן מתבצעות בדיקות כמו

האם יש **שח-מט?**

האם יש **תיקו (למשל, מחסור במהלכים חוקיים)?**

האם אחד השחקנים נכנע?

**עדכון מהלכים אפשריים:** מחשב אילו מהלכים חוקיים ניתן לבצע בכל רגע נתון. המערכת צריכה לבדוק את מצב הלוח ולהתחשב בחוקים כמו חסימת כלים, איומים על המלך, חוקים מיוחדים כמו הצרחה ועוד.

**בדיקת חוקים מיוחדים:** ניהול חוקים כמו הצרחה (הזזת המלך והצריח יחד), לקיחה "אן פאסן" (מהלך ייחודי לpawns ), וקידום pawns (כאשר ה pawn מגיע לקצה הלוח וניתן להחליפו בכלי אחר).

**מימוש מבני נתונים:** אחסון מצב הלוח והכלים באמצעות מבני נתונים מתקדמים כמו ביטבורד, טבלאות האש, ועצי חיפוש.

**עדכון מבני נתונים:** לאחר כל מהלך, המודל מעדכן את מצב הלוח, רושם את השינויים שנעשו, ומעביר את המידע לתצוגה.

**Controller:**

הבקר פועל כמתווך בין התצוגה למודל. הוא מקבל פקודות מהשחקן, מאמת אותן מול המודל, ומורה לתצוגה כיצד להתעדכן בהתאם.

**מרכיבי controller**

**יצירת הלוח**- כאשר המשחק מתחיל, הבקר יוצר את הלוח ואת הכלים בהתאם למצב הפתיחה של המשחק.

יצירת כלים -הבקר ממקם את הכלים השונים (מלך, מלכה, צריחים וכו') במיקומם ההתחלתי.

ניהול זרימת המשחק -הבקר מנהל את תורי השחקנים, מעביר את המשחק קדימה על פי המהלכים שבוצעו, ומחליט אם יש צורך לסיים את המשחק.

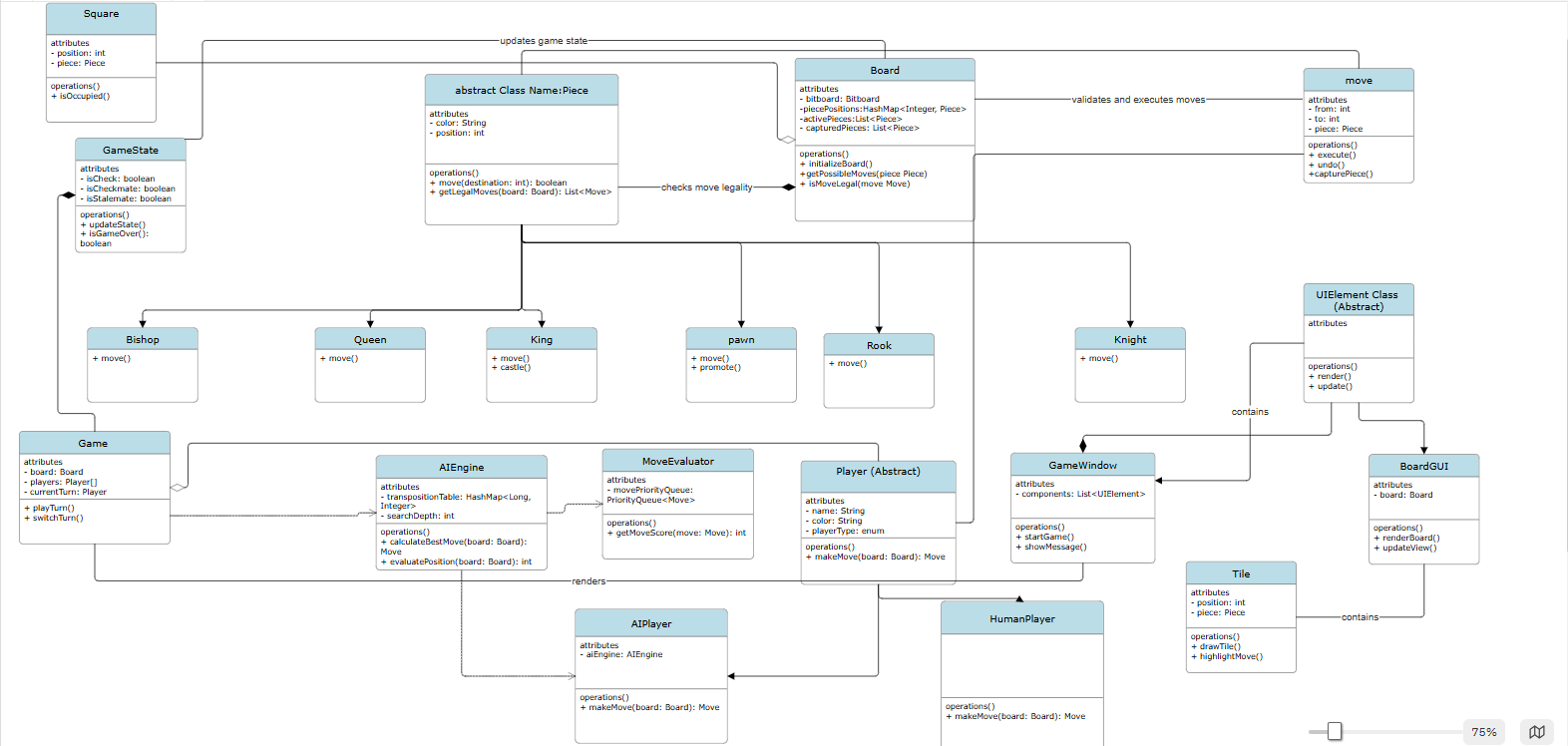
ניהול הבוט- אם המשחק כולל מנוע חכם, הבקר מפעיל אותו ומחשב מהלכים עבור המחשב. התהליך כולל

יצירת קודים מזהים ייחודיים לכל מצב לוח

חישוב המהלך הטוב ביותר באמצעות אלגוריתם חישוב

שימוש בטבלאות שמירה לחיסכון בזיכרון ולשיפור ביצועים.

**Class diagram**



**הסבר על המחלקות :**

**מחלקת Game**

מחלקה האחראית על ניהול המשחק, כולל שליטה על תורי השחקנים, ביצוע מהלכים ועדכון מצב המשחק.

**תכונות המחלקה:**

board: Board- לוח המשחק שבו מתנהלים המהלכים.

players: Player[]- מערך המכיל את שני השחקנים (לבן ושחור).

currentTurn: Player-מציין מי השחקן הנוכחי שמשחק בתור זה.

| פונקציה | הסבר |
| --- | --- |
| public Game(Board board, Player[] players) | פעולה בונה שמאתחלת את המשחק, יוצרת את הלוח והשחקנים. |
| public void playTurn() | מבצע את המהלך של השחקן הנוכחי ומעביר את התור לשחקן הבא. |
| public void switchTurn() | מחליף את תור השחקנים בין הלבן לשחור. |

**מחלקת board**

מחלקה המייצגת את לוח השחמט, שומרת על מצב הכלים ומוודאת חוקיות מהלכים.

**תכונות המחלקה:**

bitboard: Bitboard- יצוג ביטי ללוח, משמש לאופטימיזציה של חישובים.

piecePositions: HashMap<Integer, Piece>- מיפוי בין מיקומים לכלים.

activePieces: List<Piece>- רשימה של כל הכלים הפעילים על הלוח.

capturedPieces: List<Piece>- רשימה של כל הכלים שנתפסו.

gameState: GameState - מעקב אחרי מצב המשחק (שח, מט, תיקו).

**כותרת פונקציה:**

| פונקציה | הסבר |
| --- | --- |
| public Board(GameState gameState) | פעולה בונה שמאתחלת את הלוח ומחברת אותו למצב המשחק. |
| public void initializeBoard() | ממקם את כל הכלים במיקומם ההתחלתי. |
| public List<Move> getPossibleMoves(Piece piece) | מחזירה את כל המהלכים האפשריים לכלי מסוים. |
| public boolean isMoveLegal(Move move) | בודקת האם המהלך חוקי לפי חוקי השחמט. |
| public void capturePiece(Piece piece) | מטפל בלכידת כלי ומוסיף אותו לרשימת הכלים שנתפסו. |

**מחלקת GameState**

מחלקה המנהלת את מצב המשחק: שח, מט ותיקו.

**תכונות המחלקה:**

isCheck: Boolean- האם המלך בשח?

isCheckmate: Boolean- האם המשחק הסתיים במט?

isStalemate: Boolean- האם המשחק הסתיים בתיקו?

כותרת פונקציה:

| פונקציה | הסבר |
| --- | --- |
| public void updateState(Board board) | מחשבת מחדש את מצב המשחק לאחר כל מהלך. |
| public boolean isGameOver() | בודקת האם המשחק הסתיים בעקבות מט או תיקו. |

**מחלקת player**

מחלקה מופשטת המייצגת שחקן במשחק.

**תכונות המחלקה:**

name: String- שם השחקן.

color: String - צבע השחקן (לבן או שחור).

playerType: enum {HUMAN, AI} - סוג השחקן (אדם או מחשב).

**כותרת פונקציה:**

| פונקציה | הסבר |
| --- | --- |
| public abstract Move makeMove(Board board) | פונקציה מופשטת שתמומש על ידי שחקן אנושי או מחשב. |

**מחלקתHumanPlayer**

שחקן אנושי המפעיל את המהלכים דרך הממשק הגרפי.

**כותרת פונקציה**:

| פונקציה | הסבר |
| --- | --- |
| public Move makeMove(Board board) | מקבל קלט מהמשתמש ומחזיר את המהלך שבחר. |

**מחלקת AIplayer**

שחקן מחשב שמחשב מהלכים אופטימליים באמצעות AIEnginen.

**תכונות המחלקה:**

aiEngine: AIEngine- מנוע החישוב של הAI

**כותרת פונקציה:**

| פונקציה | הסבר |
| --- | --- |
| public Move makeMove(Board board) | מחשב את המהלך הטוב ביותר באמצעות ה-AI. |

**מחלקתmove**

מחלקה המייצגת מהלך במשחק.

**תכונות המחלקה:**

from: int - נקודת המוצא של הכלי.

to: int - נקודת היעד של הכלי.

piece: Piece - הכלי שמבצע את המהלך.

**כותרת פונקציה:**

| פונקציה | הסבר |
| --- | --- |
| public void execute(Board board) | מבצע את המהלך ומעדכן את מצב הלוח. |
| public void undo(Board board) | מחזיר את המצב הקודם של הלוח לפני המהלך. |

**מחלקת AIEngine**

מנוע שמחשב מהלכים אופטימליים עבור המחשב.

**תכונות המחלקה:**

transpositionTable: HashMap<Long, Integer>- טבלת עמדות להאצת חישובים.

searchDepth: int- עומק החיפוש של המחשב.

**כותרת פונקציה:**

| פונקציה | הסבר |
| --- | --- |
| public Move calculateBestMove(Board board) | מוצא את המהלך הטוב ביותר על פי חישובים. |
| public int evaluatePosition(Board board) | מעריך את מצב הלוח הנוכחי. |

**מחלקת piece**

מייצגת כל כלי שחמט במשחק.

**תכונות המחלקה:**

color: String- בצע הכלי

position: int -המיקום הנוכחי של הכלי.

**כותרת פונקציה:**

| פונקציה | הסבר |
| --- | --- |
| public abstract List<Move> getLegalMoves(Board board) | מחזירה את כל המהלכים החוקיים לכלי. |

**מחלקות הכלים (Pawn, Rook, Knight, Bishop, Queen, King)**

Pawn - כולל promote() לשדרוג רגלי.

King - כולל castle() למהלך הצרחה.

שאר הכלים מממשים move() בהתאם לכללי התנועה שלהם.