נושאים מתקדמים בתכנות

תרגיל 3

בתרגיל זה עליכם לבנות מנהל תחרות שידע להריץ תחרות בין אלגוריתמים במשחק צוללות ולדווח על תוצאות באופן מסודר. תידרש עבודה עם Threads לצורך פיצול העבודה של מנהל התחרות לניצול מקסימלי של משאבי המכונה שעליה תורץ התחרות.

כמו כן יהיו מספר שינויים באפשרויות המשחק ובחתימות של ה-interface שעליכם לממש ותידרשו להגיש אלגוריתם משחק חדש ומשופר לפי כללי המשחק המעודכנים.

עדכון כללי המשחק:

1. המשחק הינו על-פני לוח תלת מימדי בגודל דינמי של X עמודות על Y שורות ועומק של Z שכבות.
2. המידות של כלי השיט כפי שהוגדרו נשארו ללא שינוי
3. כלי השיט יכולים לשבת לאורך שורה, טור או עומק אבל לא באלכסון. הכללים לגבי מרחק בין כלי שיט חלים כעת על שלושת המימדים.
4. בכל מהלך יש לספק מיקום להפצצה שמורכב מהצירוף של X,Y,Z
5. אין צורך לתמוך בתרגיל זה בקריאת קובץ מהלכים, אבל עדיין נדרש לתמוך בקריאת קובץ לוח – פורמט קובץ הלוח יתעדכן והוא מוצג בהמשך

עדכון מחלקת הבסיס:

המחלקה האבסטרקטית IBattleshipGameAlgo (שתוצג באופן מלא ומדוייק בקובץ במקביל לקובץ התרגיל) משתנה מעט. אלגוריתם המשחק שתכתבו יידרש לממש את המתודות האבסטרקטיות שבמחלקה:

```
void setPlayer(int player); // called every time the player changes his order
void setBoard(const BoardData& board); // called once at the beginning of each new game
GameMove attack(); // ask player for his move
void notifyOnAttackResult(int player, Coordinate move, AttackResult result); // last move result
```

המחלקה BoardData:

```
class BoardData {
public:
	int rows() const;
	int cols() const;
	int depth() const;
	virtual ~BoardData() { }
	virtual char charAt(Coordinate c) const = 0;
};
```

הפונ' charAt תחזיר את ה-char שמיוצג בקורדינטה שנשלחה במידה וישנה שם ספינה שלי, אך כאשר מדובר בספינה של שחקן יריב או במשבצת ריקה הפונקציה תחזיר את התו רווח (space).

המחלקה Coordinate:

```
struct Coordinate {
	const int x, y, z;
	Coordinate (int x1, int y1, int z1): x(x1), y(y1), z(z1) { }
};
```

עליכם לממש אלגוריתם חכם אחד. האלגוריתם יניח שהספינות של היריב הן בדיוק מאותן סוגים ובאותן כמויות כמו הספינות שלו. אין צורך לבדוק זאת במנהל המשחק – ההנחה היא שהקבצים שיסופקו יישמרו על שיוויון בין השחקנים מבלי שיש צורך לוודא זאת.

בונוס: כתיבת הערת אזהרה לקובץ log במידה והלוח אינו מאוזן (לא אותם סוגים / כמויות לפי סוגים לשני השחקנים), מתוך מנהל המשחק, אבל הלוח עדיין ירוץ.

הערה כללית לגבי בונוס logger בתרגיל: מי שמחליט לממש בונוס logger, הכתיבה ל-logger הינה רק ממנהל המשחק / מנהל התחרות ולא מה-dll. כמו כן יש לדאוג שהכתיבה ל-log תתבצע מ-thread יחיד או שתהיה מסונכרנת. קובץ ה-log יהיה בתיקיה שממנה נלקחים קבצי הלוח וה-dll וייקרא game.log הכתיבה לקובץ תהיה תמיד ב-append כאשר השורה הראשונה שהתכנית תדפיס ל-log תהיה שהחלה הרצה של התכנית. כל הדפסה ל-log תהיה תמיד שורה אחת שתתחיל בתאריך ושעה ולאחר מכן ההודעה. פורמט הלוג, האם לציין severity וכו' – אתם חופשיים לבחור.

תיאור קבצי הקלט לתוכנית:

1. לוח הקרב:
מבנה לוח הקרב, חוקיותו ומדיניות דיווח השגיאות יפורטו בהמשך. עליכם להגיש לוח קרב אחד.

2. קבצי מהלכי תקיפה:
לא רלבנטיים בתרגיל זה. נא לא להגיש קבצים אלו!

3. קובץ אלגוריתם אחד (dll):
אתם אמנם מגישים קובץ אחד אך המשחק ינהל תחרות בין הרבה קבצים כפי שיפורט בהמשך.

שימו לב שהמשחק צריך לדעת לתמוך **בכל** dll הממשמש את הממשק שהוגדר, אפילו אם ה-dll לא נכתב על-ידכם, כל עוד הוא עומד בכללים שמוגדרים בתרגיל.

בבדיקה נריץ קבצים שלנו על התכנית שלכם ולכן חשוב שהתכנית תדע להתמודד עם קבצים שונים בפורמט שהוגדר ולהוציא פלט מדוייק כפי שמוגדר.

לוח הקרב:

לוח הקרב ייקרא מקובץ טקסט בפורמט הבא:

שורה ראשונה תכיל 3 מספרים מופרדים ע"י התו x (X קטן) וללא רווחים – הראשון הוא מספר העמודות, השני מספר השורות והשלישי הוא העומק, למשל:

10x20x3

בדוגמה שלמעלה ישנן 10 עמודות, 20 שורות ועומק 3.

מיד לאחר מכן תגיע שורת רווח ולאחריה יגיעו מטריצות עבור כל חתך עומק (מטריצה ראשונה לעומק 0) מופרדות בשורת רווח.

במידה ונקלט לוח קרב לא תקין יש להתעלם ממנו ולהמשיך ללוח הקרב הבא. אין צורך להדפיס למסך.
יש להימנע מבדיקת חוקיות / ניתוח וקלט של לוח נתון יותר מפעם אחת.

בונוס: הדפסת שגיאות לקובץ log כפי שתואר למעלה.

התחרות:

יש להריץ תחרות של כל ה-dll-ים שנמצאים בתיקיה שנשלחה כפרמטר command-line: כולם כנגד כולם על כל לוחות הקרב שבתיקיה כאשר על כל לוח משחקים פעמיים עם היפוך תפקידים.

אם למשל יש 5 dll-ים בתיקיה ו-5 לוחות קרב יהיו בסה"כ בדיוק 100 משחקים.

יש להריץ את התחרות במספר threads בהתאם לפרמטר שיתקבל ב-command-line ולחלק באופן יעיל וחכם את המשחקים בין ה-threads באופן שכל ה-threads יהיו מנוצלים וכן באופן שיאפשר לדווח תוצאות ביניים של הריצה באופן שיוויוני (בכל דיווח מספר המשחקים ששיחק כל אלגוריתם בשאיפה יהיה זהה).

דיווח תוצאות ביניים ותוצאות סופיות:

מנהל התחרות יחזיק thread נוסף שתפקידו לנהל את התוצאות ולדווח דיווחי ביניים.

בכל סיום "מחזור" – כלומר כאשר כל האלגוריתמים סיימו משחק (מחזור מס' 1, מס' 2 וכו') ידווח מנהל התחרות את תוצאות הביניים שהצטברו.

פורמט הדיווח המדוייק יפורסם בהמשך. בינתיים תדאגו שהנתונים יהיו קיימים ומוכנים לצורך ההדפסה.

בסיום הריצה ידווחו תוצאות הסיום בפורמט המדוייק שיפורסם בהמשך.

הרצת התכנית:

יורץ ה-exe של התכנית והוא יחפש את כל קבצי dll ב-path שבו הוא מחפש את הקובץ sboard. לפי הכללים של תרגיל 1. חובה למצוא 2 קבצי dll – לא פחות, אחרת יש להוציא הודעת שגיאה ולצאת (אבל לא עם exit).

אין חשיבות לסדר התחרות ולסדר בחירת קבצי ה-dll וקבצי לוח הקרב כל עוד ישנו כיסוי מלא כפי שנדרש.

במידה ולא מצליחים לפתוח dll יש לדלג עליו.

בונוס: כתיבת שגיאות לקובץ log.

ה-exe יקבל ב-command-line ארגומנט ראשון עבור ה-path בו יש לחפש את הקבצים שתוארו. במידה ולא נמסר פרמטר עבור ה-path התכנית תחפש את הקבצים ב-working directory. במידה וחסרים קבצים, יש להוציא הודעת שגיאה ולסיים את ריצת התכנית:

```
Wrong path: <path>
No board files (*.sboard) looking in path: <path>
Missing algorithm (dll) files looking in path: <path> (needs at least two)
```

א. אין להציג הודעות שגיאה על תקינות קובץ למסך.
ב. אין להציג הודעות שגיאה על תקינות ה-dlls למסך.

טרם תחילת המשחק יש להדפיס למסך את שתי השורות הבאות בפורמט המדוייק הבא:

```
Number of legal players: <#>
Number of legal boards: <#>
```

אין צורך לתמוך בהרצת המשחק באופן ויזואלי בתרגיל זה.

לאחר פרמטר ה-Path יכול להופיע פרמטר threads- ולאחריו מספר שייקבע את מספר ה-threads (בפורמט: -threads=<number>) מספר זה ישמש את מנהל התחרות לצורך התחרות עצמה (מלבד ה- thread הראשי שיעסוק באיסוף תוצאות והדפסת דיווחי ביניים).

אם לא סופק פרמטר threads ברירת המחדל תהיה 4 (עדיף שנתון זה לא ייכתב כמספר hard coded).

<u>בונוס</u>: קובץ קונפיגורציה עם ערכי ברירת מחדל לפרמטרים שונים מהתיקיה שבה נמצאים שאר הקבצים.

<u>ה-dll</u>:

קובץ ה-dll יכיל את מחלקת האלגוריתם שלכם וכן פונקציה גלובלית בשם GetAlgorithm שתחזיר:
std::unique_ptr<IBattleshipGameAlgo>

הפונקציה צריכה לאפשר החזרה של עותקים נוספים של האלגוריתם ככל שיידרשו.

<u>שימוש ב-smart pointers</u> - רצוי להימנע משימוש ב-new ו-delete ככל שניתן.

<u>דגשים</u>:

א. יש לעקוב אחר "שינויים לתרגיל 3" והפורום ב-moodle שהם חלק בלתי נפרד מהנחיות ודרישות התרגיל.
ב. ניתן להשתמש רק בספריות סטנדרטיות ובאלו שאושרו בתרגילים קודמים (במידת הצורך אפשר לברר בפורום –moodle אם עולה צורך בספריות נוספות).
ג. עליכם לוודא שפרוייקטים שלכם מתקמפלים ורצים בעזרת CMake לפי ההנחיות שפירסמנו במודל.

מה מגישים (ומה לא!):

א. <u>קובץ ZIP בשם</u>:
ex3_<id1>_<id2>_<id3>.zip
לדוגמא: ex3_123456789_987654321.zip (כאן יש רק 2 סטודנטים).
ב. רק סטודנט אחד מכל קבוצה צריך להגיש את התרגיל!!!
(יינתן קנס על הגשות כפולות בגין העבודה המיותרת שהדבר מייצר)
ג. <u>הקובץ יכיל</u>:
a. **כל** קבצי ה- cpp. וה-h. של התכנית כולל הקובץ IBattleshipGameAlgo.h ושל הספריה הדינאמית. **ללא קבצים מיותרים ולא נחוצים כגון stdafx.h - הקפידו להתחיל מפרוייקט ריק!**
b. קובץ בשם: "students.txt" ובתוכו היוזר האוניברסיטאי שלכם ותעודת הזהות כמו בעבר.
c. קובץ CMakeLists.txt מה-moodle – עליכם לעדכן בראש הקובץ את כל המשתנים המוגדרים באיזור לעריכה, במקומות המתאימים, ובפרט:
   i. יוזר אוניברסיטאי
   ii. ת.ז.
   iii. רשימת **כל** כל קבצי ה- cpp. וה-h. שנדרשים ל-exe.
   iv. רשימה נפרדת של **כל** קבצי ה- cpp. וה-h. שנדרשים ל-dll שלכם.
d. קבצי קלט שאיתם בדקתם את ההגשה שלכם: קובץ sboard. אחד.
e. אם בחרתם לממש את הבונוס של ה-log ו/או של קובץ הקונפיגורציה, עליכם להגיש קבצי טקסט ריקים עם השמות: bonus.log.txt, bonus.config.txt בהתאמה.
f. **נא <u>לא</u> לספק קובץ dll – אנחנו נייצר אותו מתוך הקוד שלכם!**