מכללת הדסה, החוג למדעי המחשב
מבוא לתכנות מונחה עצמים והנדסת תוכנה
סמסטר א', תשפ"ד

תרגיל 3

תאריך אחרון להגשה:

הנביאים – יום ה', 08/02/2024, בשעה 23:59

שטראוס גברים – יום ב', 05/02/2024, בשעה 23:59

מטרת התרגיל:

גם בתרגיל זה נמשיך לעסוק בתכנון ממשק ונושאים אחרים שכבר עסקנו בהם. נממש מחלקה העוסקת בהקצאת זיכרון דינמית, ועושה שימוש נכון בבנאי העתקה (Copy Constructor), אופרטור השמה (Assignment Operator) והורס (Destructor). בנוסף, בתרגיל זה נלמד להשתמש בספרייה הגרפית SFML ונוכל לתרגל שימוש בירושה.

הנחיות להתקנת הספרייה:

קיים באתר דף הנחיות להתקנת הספרייה ולהגדרת פרויקט ב־Visual Studio כך שישתמש בספרייה. עקבו אחריו בקפדנות. כדי להבטיח שהפרויקט שלכם יתקמפל גם אצל הבודק, אין מנוס מהתקנת הספרייה לפי ההוראות הנ"ל. כמו כן, לצורך הגדרת הפרויקט שלכם לתרגיל, מומלץ להיעזר בקובצי הפרויקט של דוגמאות השימוש ב־SFML שנעלה למודל (הקובץ SFMLExamples.zip).

תיאור כללי:

בתרגיל זה נבנה צייר שלבים גרפי עבור המשחק "חתול ועכבר" שיצרנו בתרגיל 2. המשתמשים יוכלו לבחור היכן למקם את הדמויות השונות במשחק ואיך ייראה לוח המשחק. לאחר ציור הלוח, תהיה אפשרות לשמור אותו בקובץ (ובקובץ שנוצר נוכל להשתמש בהמשך כקלט לפרוייקט!).

פירוט הדרישות:

<u>הצייר</u> יהיה חלון בודד, ובו כל אפשרויות הציור. בפתיחה הראשונה של הצייר יהיו שתי אפשרויות:

1. אם קובץ הקלט "Board.txt" (עליו נרחיב בהמשך) עדיין לא קיים, לפני שיופיע משטח העריכה הנקי, תוצג בטרמינל בקשה להקליד את גודל הלוח (גובה ורוחב) הרצוי.

2. אם קובץ הקלט "Board.txt" קיים, יוצג משטח עריכה שמכיל את האובייקטים כפי שנטענו מהקובץ. באופן כללי אין צורך לבדוק את תקינות הקובץ וניתן להניח שאם הקובץ קיים, הוא אכן קובץ תקין. בהמשך נרחיב לגבי הפורמט של אותו קובץ.

הערה: כנזכר לעיל, בהרצת התוכנית – הקובץ לא בהכרח קיים. הוא יווצר על ידי התוכנית (בשמירה), ולכן הוא לא יהיה מלכתחילה בתיקיית resources (ולא נוסיף את השם שלו ל־CMakeLists של תיקייה זו; כן נשים בה קובצי תמונה וכדומה). אם הוא קיים (למשל, אחרי שמירת הלוח) – ניתן יהיה למצוא אותו בתיקיה בה נמצאת התוכנית המקומפלת (בקונפיגורציה הרגילה, זו התיקייה out\build\x64-Debug מתחת לתיקיית הפרויקט, כלומר בתוך התיקייה שבה קובץ ה־CMakeLists הראשי).

<u>חלון התכנית</u> יורכב מהחלקים הבאים:

1. שטח שישמש כמשטח העריכה, בו "נצייר".
2. תפריט שימוקם **בצד שמאל** של ה"דף".

<u>התפריט</u>, שכאמור ימוקם בצד שמאל, מאפשר למשתמשים לבחור את סוג האובייקט שירצו למקם על הלוח ואת הפעולות הרצויות.

התפריט יכיל את הכפתורים הבאים:

1. **כפתור עבור כל אחד מסוגי האובייקטים שהוגדרו במשחק** וכפי שהם מפורטים שוב להלן, לבחירת הצורה לציור. אחרי הלחיצה על אחד מכפתורי האובייקטים בתפריט, כל לחיצה על משטח העריכה תמקם במשבצת המתאימה את האובייקט, אם אין מגבלה אחרת (ראו בהמשך). נדגיש: כל לחיצה עם העכבר על משטח העריכה תניח עליו אובייקט חדש מהסוג שנבחר בתפריט הכלים, כלומר אין צורך ללחוץ על הכפתור לפני כל הוספת אובייקט, מספיק לבחור פעם אחת ואז אפשר להניח מספר אובייקטים חדשים מאותו הסוג.
2. **כפתור למחיקה** כדי למחוק אובייקט קיים. לאחר לחיצה על כפתור המחיקה בתפריט, לחיצה על אובייקט שנמצא על משטח העריכה תמחק אותו. גם כאן, כמו בכפתורים מסעיף 1, אין צורך ללחוץ על כפתור המחיקה לפני כל מחיקת אובייקט, מספיק ללחוץ על כפתור המחיקה פעם אחת ואז אפשר למחוק מספר אובייקטים מהלוח.
3. **כפתור "דף חלק"** כדי להתחיל את העריכה מחדש. כמובן שזה דורש ניקוי של משטח העריכה מכל האובייקטים שנוצרו לפני כן, אבל מלבד זה גם נבקש מחדש מהמשתמשים את גודל משטח העריכה (גודל השלב). לשם כך, נסגור את החלון הגרפי ונציג בחלון הטרמינל בקשה מהמשתמש להקליד: 1. את מספר השורות בשלב (כלומר גובה השלב), 2. את מספר העמודות בשלב (כלומר רוחב השלב). כמובן לאחר מכן ניצור את החלון הגרפי מחדש כך שיתאים לגודל השלב ומשטח העריכה כפי שהתקבלו

בטרמינל. שימו לב שכדי שבכלל יהיה לתוכנית חלון טרמינל, היא חייבת להיות מוגדרת כ־Console Application ולא כ־Win32 Application. המשמעות היא שב־CMaeLists הראשי צריך לשנות את השורה:

add_executable (${CMAKE_PROJECT_NAME} WIN32)

להיות:

add_executable (${CMAKE_PROJECT_NAME})

כלומר, למחוק את המילה "WIN32".

4. **כפתור שמירה**, לשמירת השלב לקובץ.

<u>האובייקטים האפשריים</u> הם: עכבר, חתול, קיר, גבינה, מתנה, דלת, מפתח.

נשאיר לבחירתכם מה תהיה ההתנהגות המדויקת כאשר המשתמש מנסה לשים יותר מעכבר אחד, אבל כן נגדיר בפירוש מגבלה אחת: עכבר יכול להופיע רק פעם אחת בשלב, כדי שקובץ התוצאה יוכל לשמש כקלט למשחק. (כנראה שיש שתי אפשרויות להתנהלות במקרה שהדמות של העכבר כבר מופיעה: 1. לא ניתן יהיה לשים עכבר במקום אחר לפני שמוחקים את הדמות הקיימת, כלומר הלחיצה לא תעשה כלום. 2. הדמות הראשונה תימחק אוטומטית כששמים עכבר במקום אחר).

באחריות המשתמש לסדר נכון את הלוח על מנת לאפשר מעבר שלב. למשל, לוודא שאין גבינה שמוקפת בקירות ואי אפשר להגיע אליה, שמספר המפתחות הוא לפחות כמספר הדלתות שחובה לפתוח כדי לנצח (דלתות שיש מאחוריהן גבינה) כך שבאמת ניתן לפתוח את כולן ולאסוף את כל פיסות הגבינה, וכדומה.

הצגת האובייקטים על משטח העריכה יכולה להיות זהה להצגתם בתרגיל 2, כך:

% – עכבר.

^ – חתול.

– קיר.

* – גבינה.

$ – מתנה.

D – דלת.

F – מפתח.

אבל ניתן כמובן להציג אותם גם בכל דרך אחרת. (אתם משתמשים בספריה גרפית, תחשבו בגדול! הניקוד בהתאם)

<u>הקובץ בו נשמור את הנתונים</u> יהיה קובץ קבוע בשם Board.txt, שאותו נטען אוטומטית (אם קיים) בתחילת ריצת התוכנית וכל לחיצה על כפתור השמירה תשמור את המידע אליו. הצייר שלנו יתמוך בשמירה לקובץ משחק המכיל שלב אחד בלבד, שהוא למעשה המסך שהמשתמש ערך.
פורמט הקובץ נתון לבחירתכם (ויכול להיות זהה לפורמט שאותו בחרתם בתרגיל 2. בכל מקרה, **עליכם לתעד בקובץ ה־Readme את הפורמט שבחרתם**, ולא רק להפנות ל־Readme של תרגיל 2). האובייקטים במשחק יתוארו בקובץ באמצעות אותם התווים בהם השתמשנו בתרגיל 2 ('%' – עכבר, '^' – חתול, '#' – קיר, '*' – גבינה, '$' – מתנה, 'D' – דלת, 'F' – מפתח, ' ' (התו רווח) – אריח ריק), כך תוכלו לשחק בשלב שבניתם, עם התאמות מינימליות בקובץ, וללא שום שינוי בקוד שלכם מתרגיל 2. שימו לב, כאמור לעיל, הקובץ הנשמר יכיל רק שלב אחד.

<u>מחלקת מבנה נתונים</u> על מנת לתרגל שימוש נכון בהקצאות זיכרון דינמיות, ובפרט הניהול שלהן בזמן העתקה, השמה והריסה, נבנה בתרגיל מחלקה בשם Row שמממשת מערך דינמי וכחלק מכך אחראית לניהול הזיכרון עבורו (הקצאות, שחרורים וכו'). **במחלקה זו נשתמש עבור ייצוג השורות בלוח**. כלומר, לוח המשחק ייוצג ע"י וקטור שכל איבר בו הוא שורה (Row). כמובן, במימוש מחלקה זו אין להשתמש במחלקות מוכנות (כגון std::vector או std::list) שתייתרנה את הצורך בניהול הזיכרון. את השורות עצמן (שמיוצגות ע"י המערך הדינמי שמימשתם), כפי שאמרנו - מותר (ומומלץ) להחזיק בתוך וקטור.

דרישות למחלקה זו (בהנחה שלמחלקה שמייצגת תא/אריח בלוח קראנו Tile):

במחלקה זו נגדיר בנאי המקבל את גודל מבנה הנתונים המבוקש וערך (Tile) שאליו יאותחלו כל איברי מבנה הנתונים. בנאי זה ישמש גם כבנאי ברירת המחדל. ברירות המחדל של הבנאי יהיו גודל 0 וערך ברירת המחדל של Tile. כלומר, החתימה של הבנאי תהיה:

explicit Row(int size = 0, const Tile& value = Tile());

על הבנאי הזה להיות explicit כדי למנוע המרה לא רצויה.

כצפוי, במחלקה זו נידרש לטפל בהעתקה (כלומר, בנאי העתקה ואופרטור השמה) והריסה כראוי:

Row(const Row& other);

Row& operator=(const Row& other);

~Row();

פונקציית גישה at שתקבל אינדקס כ־int ותחזיר הפניה לאיבר במיקום הזה במבנה הנתונים. אין דרישה לוודא שהאינדקס תקין (האחריות על הבדיקה הזו מוטלת על הקוד שקורא לפונקציה, למשל על ידי בדיקה מוקדמת של האינדקס מול התוצאה של פונקציית size()). נגדיר שתי גרסאות של הפונקציה, אחת שתשמש לקריאה בלבד (const) ואחת שתשמש גם לכתיבה:

const Tile& at(int index) const;
Tile& at(int index);

void push_back(const Tile& val) – מוסיפה איבר לסוף השורה (כלומר, מגדילה את גודל השורה ב-1, ומטפלת בניהול הזיכרון, ואז מכניסה את האיבר החדש לסוף השורה).

int size() const – מחזירה את גודל מבנה הנתונים.

bool empty() const – מחזירה האם מבנה הנתונים ריק.

הערות:

- גם אם החלטתם לשנות את הפורמט שבו השתמשתם בתרגיל 2, אין צורך לעדכן את תרגיל 2.
- כדי לגרום לכך שהמחלקות שניצור תהיינה משמעותיות, השימוש ב־SFML צריך להיות מצומצם רק לפעולת הציור עצמה, **כלומר האובייקטים לא יירשו ממחלקות של SFML ולא יכילו אובייקט כזה.** רק בפונקציית הציור של האובייקט ניצור את האובייקט המתאים של SFML לפי הפרמטרים השמורים באובייקט שלנו, ונצייר אותו (נבהיר כאן שהכוונה היא רק לאובייקטים שמייצגים את המחלקה, כמו sf::Text, sf::Sprite או שאר מיני ה־sf::Shape. מותר **<u>ואף רצוי</u>** להחזיק members מטיפוס צבע, Font, Vector2f, או Texture (או רפרנס/מצביע לכאלה). נשים לב שאובייקטים מסוג Font או Texture נקפיד לטעון במינימום הנדרש, ובלי העתקות, לכן לא נעשה אותם members של האובייקטים המונחים על הלוח, אלא של אובייקט אחראי כלשהו).
- ניתן לכם פה כר נרחב להחלטות אישיות על עיצוב המחלקות והממשק, בפרט בדברים שקשורים לדרך הנכונה לניהול התקשורת עם הספרייה הגרפית והאירועים השונים.
- אם נרצה שיהיה ברור מה הולך לקרות כשהעכבר עובר על המשטח במצב ציור, נוכל לעשות זאת בשתי דרכים. האפשרות האידיאלית היא לצייר כבר את האובייקט כפי שהוא הולך להיות מונח על הלוח (אם אפשר להניח אותו במקום הנוכחי) אבל "שקוף למחצה" והוא יזוז יחד עם העכבר (עד שהלחיצה תקבע אותו במקום מסוים ואז הוא לא יהיה שקוף למחצה יותר). אפשרות פשוטה יותר היא לצייר מראש את משטח הציור עם רשת של משבצות, כך שיהיה ברור כשלוחצים באיזו משבצת ימוקם האובייקט החדש ונדגיש את הריבוע שעליו העכבר מצביע כרגע (ושוב, נבדיל בין מצב שבו ניתן לצייר על הריבוע הזה או שלא ניתן).

  ובאופן דומה במצב מחיקה. אם נרצה שיהיה ברור איזה אובייקט עומד להימחק, כשהעכבר רק עובר (עוד לפני שיש לחיצה), נוכל להדגיש את הריבוע או האובייקט שהעכבר עומד מעליו, ונבדיל בין מצב שבו אנחנו מעל אובייקט או מעל ריבוע ריק (ואין מה לבחור).

- ניתן להשתמש בירושה כדי להגדיר את מחלקות האובייקטים השונות (שימו לב שהתיכון במקרה הזה צריך להתאים לעורך ולא למשחק, כלומר אין חלוקה פנימית בין אובייקטים זזים לאובייקטים שאינם זזים, כי בעורך אף אובייקט לא זז).
- אם השתמשנו בירושה – נקפיד על ממשק אחיד בין מחלקת הבסיס והמחלקות היורשות.
- שימו לב מה צריך להיות מחלקות שונות ומה יכול להיות רק הבדל במאפיינים הנשמרים.

הערות כלליות:

הרבה פרטים הוגדרו לעיל במדויק, אולם תמיד, בכל פרויקט מספיק מורכב, יש התנהגויות שלא מוגדרות במדויק על ידי הדרישות. אפשר להתייעץ במידת הצורך, אבל ההנחייה הכללית היא שעל דברים שלא נאמרו בהגדרות אתם מוזמנים להחליט בעצמכם מה הדרך הטובה יותר לביצוע הפעולה או לפתרון הבעיה (כלומר, טובה יותר מבחינת משחקיות, או סבירה מבחינת משחקיות וקלה מבחינה תכנותית :)). על כל החלטה כזו, נבקש שתתעדו את הדברים בקובץ ה־README, ולהסביר מדוע בחרתם בהחלטה הזו.

תיעוד המחלקות:

אין צורך לתעד קוד שהוא מובן מאליו (למשל, בגלל שמות משמעותיים או אופרטור שמשמעותו ברורה), זה מיותר גם בהצהרות וגם במימושים. יש להוסיף תיעוד בקוד במקרים הבאים:

- תיעוד כללי (וקצר) על ממשק המחלקה (מטרת המחלקה עצמה ופונקציות ציבוריות למיניהן)
- בפונקציות פחות טריוויאליות (לא setToDefault, נניח)
- כשיש לוגיקה לא טריוויאלית (במימוש)
- במקרים של קוד לא ברור מאליו ו"מפתיע" (למשל, העברת ערך לא חוקי בכוונה כדי לקבל כתוצאה את ברירת המחדל)

קובץ ה־README

יש לכלול קובץ README שיקרא README.docx, README.doc או README.txt (ולא בשם אחר). הקובץ יכול להיכתב בעברית ובלבד שיכיל את הסעיפים הנדרשים.

קובץ זה יכיל לכל הפחות:

1. כותרת.
2. פרטי הסטודנט: שם מלא כפי שהוא מופיע ברשימות המכללה, ת"ז.
3. הסבר כללי של התרגיל.
4. רשימה של הקבצים שיצרנו, עם הסבר קצר (לרוב לא יותר משורה או שתיים) לגבי תפקיד הקובץ.

5. מבני נתונים עיקריים ותפקידיהם.
6. אלגוריתמים הראויים לציון.
7. **תיכון (design): הסבר קצר מהם האובייקטים השונים בתוכנית, מה התפקיד של כל אחד מהם וחלוקת האחריות ביניהם ואיך מתבצעת האינטראקציה בין האובייקטים השונים.**
8. באגים ידועים.
9. הערות אחרות.

יש לתמצת ככל שניתן אך לא לוותר על אף חלק. אם אין מה להגיד בנושא מסוים יש להשאיר את הכותרת ומתחתיו פסקה ריקה. **נכתוב ב־README כל דבר שרצוי שהבודק ידע כשהוא בודק את התרגיל.**

אופן ההגשה

הקובץ להגשה: ניתן ליצור בקלות קובץ zip המותאם להגדרות ההגשה המפורטות להלן ישירות מתוך VS, כפי המוסבר תחת הכותרת "יצירת קובץ ZIP להגשה או לגיבוי" בקובץ "הנחיות לשימוש ב־Visual Studio 2022". אנא השתמשו בדרך זו (אחרי שהגדרתם כראוי את שמות הצוות ב־MY_AUTHORS: **נשים את שמות המגישים בתוך המרכאות, נקפיד להפריד בין השם הפרטי ושם המשפחה בעזרת קו תחתי ואם יש יותר ממגיש אחד, נפריד בין השמות השונים בעזרת מקף (מינוס '-')**) וכך תקבלו אוטומטית קובץ zip המותאם להוראות, בלי טעויות שיגררו אחר כך בעיות בבדיקה.

באופן כללי, הדרישה היא ליצור קובץ zip בשם oop1_exN-firstname_lastname.zip (או במקרה של הגשה בזוג – oop1_exN-firstname1_lastname1-firstname2_lastname2.zip), כשהקובץ כולל את כל קובצי הפרויקט, למעט תיקיות out ו־vs. כל הקבצים יהיו בתוך תיקייה ראשית אחת.

את הקובץ יש להעלות ל־Moodle של הקורס למשימה המתאימה. בכל מקרה, **רק אחד** מהמגישים יגיש את הקובץ ולא שניהם.

הגשה חוזרת: אם מסיבה כלשהי החלטתם להגיש הגשה חוזרת יש לוודא ששם הקובץ זהה לחלוטין לשם הקובץ המקורי. אחרת, אין הבודק אחראי לבדוק את הקובץ האחרון שיוגש.

כל שינוי ממה שמוגדר פה לגבי צורת ההגשה ומבנה ה־README עלול לגרור הורדת נקודות בציון.

מספר הערות:

1. נשים לב לשם הקובץ שאכן יכלול את שמות המגישים.
2. נשים לב לשלוח את תיקיית הפרוייקט כולה, לא רק את קובצי הקוד שהוספנו. תרגיל שלא יכלול את כל הקבצים הנדרשים, לא יתקבל וידרוש הגשה חוזרת (עם כללי האיחור הרגילים).

המלצה כללית: אחרי הכנת הקובץ להגשה, נעתיק אותו לתיקייה חדשה, נחלץ את הקבצים שבתוכו ונבדוק אם ניתן לפתוח את התיקייה הזו ולקמפל את הקוד. הרבה טעויות של שכחת קבצים יכולות להימנע על ידי בדיקה כזו.

בהצלחה!