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

תרגיל 2

תאריך אחרון להגשה:

יום ב', 19/11/2018 בשעה 23:59

מטרת התרגיל:

בתרגיל זה נתרגל תכנון ממשק (Interface designing), שימוש במחלקות מורכבות (מחלקות המכילות אובייקטים של מחלקות אחרות), העברת מידע בין אובייקטים, וכתלות בהתקדמות ההרצאות גם נושאים של שימוש נכון בהפניות (References) וב־const.

תיאור כללי:

בתרגיל זה נבנה תכנית המשתמשת במספר אובייקטים. לשם ביצוע המשימה האובייקטים יצטרכו להעביר מידע ביניהם. התוכנית אותה עליכם לממש היא משחק Bomberman פשוט בעל שני מסכים לפחות. בהמשך יפורטו הכללים המדויקים של המשחק כפי שתממשו אותו.

עלילת הרקע מתוארת בוויקיפדיה:

בומברמן הוא רובוט הרכבה העובד במפעל פצצות, והוא מרכיב הפצצות הטוב ביותר במפעל. בומברמן משתעמם מהכנת הפצצות במפעל יום אחר יום ללא הפסקה. בעקבות שמועה שרובוט שיצליח לצאת מהמפעל התת-קרקעי יקבל משאלה, הוא מתחיל במסע ברחבי המפעל כדי שיוכל לבקש משאלה להפוך לבן אדם. המכשול העיקרי בדרכו הוא התנגדותם של שומרי המפעל, העושים כל שביכולתם כדי שהמפעל ימשיך לתפקד כרגיל.

כדי להימלט, על בומברמן למצוא את דרכו במבוך תוך כדי הימנעות מאויבים. הדלתות המובילות לחדרי מבוך נוספים (השלבים הבאים במשחק) מוסתרות על ידי סלעים, אותם בומברמן צריך לפוצץ. ישנם חפצים שיכולים לשפר את הפצצות שבומברמן מייצר, כגון חפצים העוזרים ליצור פצצות חזקות יותר או פצצות המופעלות בשלט רחוק.

המשחק זמין ברשת בקישור הבא: https://www.retrogames.cz/play_085-NES.php

גרסה אחרת שלו זמינה כאן: http://www.crazygames.com/game/bomber7io

אנחנו נממש גרסה פשוטה יותר של המשחק, כפי שיפורט בהמשך.

אתם צריכים לחשוב על התיכון (Design) המתאים מבחינת אלו מחלקות צריכות להיות, חלוקת האחריות ביניהן, אלו פונקציות יהיו בכל אחת וכו'. שימו לב שאתם נדרשים לתאר את התיכון ולהסביר אותו בקובץ ה־Readme (ראו בסוף הקובץ, בחלק המתאר את קובץ ה־Readme).

פירוט הדרישות:

האובייקטים במשחק:

רובוט – זו הדמות שמייצגת את השחקן. מטרת המשחק היא להביא את הרובוט אל דלת היציאה במספר צעדים מוגבל ובלי להיתפס על ידי השומרים.

שומרים – דמויות הנשלטות על ידי המחשב. מנסים למנוע מהשחקן להגיע אל דלת היציאה. אם הם נוגעים בשחקן – השחקן נפסל ("יורד לו חיים").

דלת היציאה – המטרה הנכספת. שלב מסתיים בהצלחה בהגעה של הרובוט אל דלת היציאה.

קירות – אף דמות לא יכולה לעבור דרך קיר. גם פצצות אינן משפיעות על קירות.

סלעים – מכשולים נוספים העומדים בדרכם של הדמויות. אף דמות לא יכולה לעבור סלע, אבל הפצצות שהשחקן מניח מפוצצות את הסלעים ואז המעבר פנוי לכולם.

פצצות – לשחקן מלאי בלתי מוגבל של פצצות, אותן הוא יכול להניח. כל פצצה שהשחקן מניח מתפוצצת אחרי ארבעה תורים ומשמידה כל דמות (רובוט או שומר) או סלע הנמצאים במשבצת שלה או באחת מארבע המשבצות הסמוכות לה (מעלה, מטה, ימינה ושמאלה).

מהלך המשחק ותנועת הדמויות:

לשם פישוט, המשחק לא מתנהל "בזמן אמת" לפי שעון, כמו המשחק הרגיל, אלא לפי תור. בכל תור דמות יכולה לזוז לאחד מארבעת הכיוונים (אך לא באלכסון). דמות זזה בכל תור רק משבצת אחת.

נבחין בין המקרים הבאים לגבי התא המבוקש:

- אם התא המבוקש חסום (יש בו קיר או סלע), התנועה לא תתבצע. מה כן יקרה? נבדיל בין השחקן לבין השומרים, אם השחקן ניסה לזוז בכיוון המכשול ונחסם, התנועה לא תתבצע אך גם התור נשאר שלו, עד שהוא יזוז או עד שהוא יוותר במפורש על התור שלו (ראו בהמשך). לעומת זאת, אם שומר ניסה לזוז ונחסם, הוא מפסיד את התור, והתור עובר לדמות הבאה לפי הסדר.
- גבולות המסך גם הם משמשים כ"קירות", למשל, גם אם ניתן להגיע לקצה השמאלי של המסך בשורה מסוימת, ניסיון לנוע שמאלה נחשב כמו ניסיון כניסה לתא חסום (עם הכללים דלעיל).

- אם השחקן נע לתא שיש בו שומר, המהלך חוקי (הוא מתבצע), אולם הוא "נפגע" מיד.
- כצפוי, גם אם שומר נע לתא שבו נמצא השחקן, השחקן "נפגע".
- אם שומר נע לתא שיש בו שומר, המהלך חוקי. כלומר, שני שומרים יכולים להיות בתא אחד.
- ניתן להיכנס לתא שיש בו פצצה או פיצוץ (אלא שבמקרה של פיצוץ, יש פגיעה מיידית).
- אם השחקן נע לתא שבו נמצאת הדלת, השלב מסתיים.
- הבדל נוסף בין השחקן לשומרים, השחקן יכול לבחור לוותר על תורו (ראו להלן), אולם שומר לא יכול "לבחור" לוותר על תורו, אלא רק להפסיד את התור במקרה שניסה לזוז לתא חסום, כנ"ל.
- השחקן יכול לבחור להניח פצצה בתורו במקום לנוע. לא ניתן להניח פצצה בתא שכבר יש בו פצצה.
- הנחת פצצה או ויתור על תור לא נספרים במספר הצעדים של השלב, כלומר בתור כזה מספר הצעדים הנותר נשאר ללא שינוי.
- אחרי התור של השחקן והשומרים, מגיע ה"תור" של כל אחת מהפצצות (לבחירתכם אם זה תלוי בסדר שבו השחקן הניח אותן או שזה לפי סדר המיקום שלהן בלוח, העיקר שיהיה סדר אחיד וקבוע). ב"תור" של הפצצה, המונה שלה קטן ב־1, ואם הוא הגיע ל־0 היא מתפוצצת. שימו לב שכשהשחקן מניח פצצה, יש לה תור כמעט מיד, ולכן, כדי שהיא תתפוצץ רק כעבור ארבעה תורים של השחקן, כשהיא נוצרת צריך להתחיל ממונה של 5. כך, כשמגיע התור הראשון שלה המונה הופך ל־4.
- כל דמות שנמצאת במשבצת כאשר קורה בה פיצוץ או נכנסת למשבצת שיש בה פיצוץ (כי הרי הפיצוץ נשאר כך למשך תור אחד) – נפגעת מיד. שומר פשוט נעלם מהשלב. הטיפול בשחקן מעט מורכב יותר ומפורט בהמשך.

מקשי השחקן:

השחקן מזיז את הדמות שלו בעזרת מקשי החיצים (ראו הסבר בהמשך), מניח פצצה בלחיצה על המקש b, או לוחץ על מקש הרווח כדי לוותר על התור.

תזוזת השומרים:

השיטה להזזת השומרים נתונה לבחירתכם. מספר נקודות מציון התרגיל מוקדשות לנושא זה וככל שהשיטה מוצלחת יותר, הניקוד בסעיף זה יעלה. תזוזה אקראית היא האפשרות הפשוטה כאן, אולם

היא גם "מתומחרת" בהתאם. בכל מקרה, עליכם לציין בקובץ ה־Readme מה השיטה שהחלטתם לממש ולהסביר את הבחירה בה, איך היא עובדת וכדומה.

סיום השלב:

כאמור, כאשר השחקן "עובר בדלת", הוא סיים בהצלחה את השלב הנוכחי, והתכנית צריכה לטעון את השלב הבא (אם קיים כזה). אם אין שלב נוסף, השחקן ניצח במשחק.

לעומת זאת, אם השחקן "בזבז" את כל הצעדים המוקצים לשלב הזה (כפי שנקבע בקובץ, כמו שמוסבר בהמשך) ועדיין לא סיים אותו, יורד לו "חיים" והשלב מתחיל מחדש.

אם השחקן "נפגע" על ידי שומר או פצצה, יורד לו ממספר ה"חיים" שנותרו לו. אם לא נשארו לו עוד חיים, המשחק מסתיים בכישלון של השחקן. אם נשארו חיים, השלב לא מתחיל מחדש, הוא ממשיך מאותו מקום, אבל את הדמויות (השחקן והשומרים שנותרו) נציב מחדש במקומות שבהם הם היו בתחילת השלב (כדי לא להגיע למצב שהשחקן נפגע שוב ושוב על ידי אותו שומר, למשל). בנוסף להצבה מחדש הזו, אם יש פצצה בתא של השחקן או באחד התאים שלידו, כך שאם היא תתפוצץ היא תפגע בו (גם אם עוד נותרו מספר תורים עד שהיא תתפוצץ), נמחק אותה מהלוח.

מספר החיים ההתחלתי לשחקן יהיה קבוע ל־3 (תזכורת: אין להשתמש במספרים מפורשים, literals , ישירות בקוד hard-coded, אלא להשתמש בקבוע כלשהו לאתחול).

ניקוד:

כל שומר שמסולק, מזכה את השחקן ב־5 נקודות כפול מספר השומרים שהיו בתחילת השלב הנוכחי. למשל, בשלב שמתחיל עם שלושה שומרים, הניקוד על כל שומר (גם האחרון שנשאר) הוא 15 נקודות.

סיום השלב מזכה בבונוס של 20 נקודות כפול מספר השומרים באותו אופן, כלומר בדוגמה הקודמת, סיום השלב מזכה את השחקן ב־60 נקודות, לא משנה כמה שומרים נשארו בסוף השלב.

מסכי המשחק:

מסכי המשחק יוגדרו בקובץ טקסט בשם Board.txt, והתכנית תקרא אותם מקובץ זה.

בקובץ הזה "נצייר" את המסכים באופן הבא:

השורה הראשונה לפני כל "ציור" שלב בקובץ תכיל שני מספרים. הראשון יהיה מספר המגדיר את גודל הלוח לשלב הנוכחי. כלומר אם המספר הינו N אזי השלב יהיה בגודל של N על N משבצות (כלומר, N שורות בנות N תווים כל אחת) וזה מספר השורות (והעמודות בכל שורה) שאנחנו מצפים לקרוא מהקובץ עבור השלב הזה. המספר השני יהיה מספר הצעדים המוקצה לשלב הזה.

השורות הבאות (N שורות, כאמור) מתארות את מבנה השלב ההתחלתי, באופן הבא:

רווח משמעותו משבצת ריקה.

– קיר.

/ – הרובוט, כלומר דמות השחקן.

! – שומר.

D – דלת היציאה.

@ – סלע.

למען הסר ספק, בקובץ אין פצצות. פצצות מונחות על ידי השחקן במהלך המשחק.

כל השלבים נמצאים באותו הקובץ, ובין שלב לשלב נשים שורה ריקה להפרדה.

כפי שנאמר לעיל, אתם נדרשים ליצור לפחות שני מסכים, אבל שימו לב שבקוד שלכם לא תהיה הגבלה על מספר השלבים האפשרי. אם מישהו מוסיף שלב לקובץ הזה, התוכנית אמורה לקרוא אותו ללא בעיות ולהמשיך כך את המשחק לשלב נוסף. כמו כן, התוכנית צריכה להתמודד גם אם ניתן לה קובץ שלבים ממשחק אחר.

ניתן להניח תקינות של הקובץ.

הצגת לוח המשחק:

אחרי כל תור (כולל תור של שומר או פצצה) יש להציג למסך את לוח המשחק, מצב הניקוד, את החיים והצעדים הנותרים ואת מספר השלב. כלומר, כל שינוי מעדכן את התצוגה. בסיום המשחק, נציג הודעה לשחקן אם הוא ניצח או הפסיד, בהתאם לכללים שהוגדרו לעיל.

ציור הדמויות והאובייקטים השונים זהה לצורה שבה הם מופיעים בקובץ השלבים, בתוספת הנקודות הבאות:

- ציור הפצצה נעשה בעזרת מספר המציין את מספר התורים שנותר עד שהיא מתפוצצת, כלומר אנחנו נראה ספירה לאחור, 4, 3, 2, 1.
- כאשר הפצצה מתפוצצת, נציג למשך תור אחד כוכביות (*) במקומות שבהם היא מפוצצת (התא שלה וארבעת התאים הסמוכים מעליה, מתחתיה, מימין לה ומשמאל לה, אבל רק אם אין שם קיר).
- כאשר קיימים מספר אובייקטים שונים באותו המקום, נעדיף להציג שומר מאשר להציג פצצה. אם שחקן נמצא בתא שיש בו פצצה (כולל המצב שבו הוא הניח פצצה כעת), נסמן אותו כ-% במקום / הרגיל.

ניקוי לוח המשחק:

כדי לאפשר הצגה ברורה של העדכונים, נצטרך לנקות את המסך בין הדפסה להדפסה. זאת נשיג על ידי הפקודה ("system("cls (שכלולה בספריית cstdlib). כדאי לציין ש־system משתמשת בפקודות המערכת, ולכן הפקודה המסוימת הזו תעבוד רק בסביבת Windows ולא במערכות הפעלה אחרות.

אפשרות נוספת היא להשתמש בקוד הבא (שגם הוא ספציפי ל־Windows):

```
SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), {0, 0});
```

בשביל הקוד הזה צריך לעשות include לקובץ Windows.h ומומלץ מאוד לעשות זאת בקובץ cpp נפרד. הקריאה הזו מחזירה את הסמן לתחילת המסך, ולכן כל מה שנדפיס כעת משכתב את מה שכבר מופיע. אם אתם בוחרים בדרך הזו, שימו לב לוודא שבכל השורות אתם מדפיסים את כל רוחב השורה (כולל רווחים) כדי למחוק את הטקסט שהיה שם קודם.

שימוש במקשי החיצים:

עד היום הכרנו איך לקבל מהמשתמש קלט שמכיל תווי ASCII, כאלה שיש להם ייצוג מודפס על המסך. כמו כן, שיטות הקלט הרגילות (בין אם זו cin, scanf או אפילו getc) דורשות מאתנו להמתין עד שהמשתמש ילחץ על מקש ה־Enter לפני שהן מתחילות לקבל את הקלט. המגבלה הראשונה מונעת מאתנו להשתמש במקשי החיצים בכלל. המגבלה השנייה מונעת מאתנו לקבל את הקלט מהמשתמש (לא משנה באלו מקשים נבחר לשם כך) בלי לחייב אותו ללחוץ Enter בכל צעד, דבר שאיננו סביר במשחק כמו זה.

כדי להשתמש במקשי החיצים, וכדי לקבל את המקש שנלחץ גם בלי המתנה ל-Enter, אנחנו נאלצים לפנות לספריה שאיננה חלק מהסטנדרט של C או C++. הפעם בחרנו להשתמש בספריית conio.h. הסתכלו בקובץ הדוגמה המצורף, ConioExample.cpp, כדי לראות איך משתמשים בה לצורכנו. שימו לב שלמקשים מיוחדים, כאלה שאין להם ערך ASCII, כמו מקשי החיצים והפונקציות, מוחזרים שני ערכים בזה אחר זה וצריך לקרוא לפונקציה ()getch_ פעמיים לשם זיהויים, בפעם הראשונה מוחזר 0 או 224 (תלוי במקש/במימוש/במערכת ההפעלה) ובפעם השנייה מוחזר הקוד המזהה את המקש המסוים. חייבים לזהות את מה שמוחזר בפעם הראשונה ולזכור זאת, כפי שתראו בקובץ הדוגמה, מכיוון שמה שמוחזר בפעם השנייה זה קוד שבמצב רגיל מדבר על מקש אחר לחלוטין (H, K, M או P, במקרה של מקשי החיצים).

הערה כללית:

הרבה פרטים הוגדרו לעיל במדויק, אולם תמיד, בכל פרויקט מספיק מורכב, יש התנהגויות שלא מוגדרות במדויק על ידי הדרישות. אפשר להתייעץ במקרה 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 :)).

קובץ ה־README:

יש לכלול קובץ README שיקרא README.doc, README.docx 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. **תיכון (design): הסבר קצר מהם האובייקטים השונים בתוכנית, מה התפקיד של כל אחד מהם וחלוקת האחריות ביניהם ואיך מתבצעת האינטראקציה בין האובייקטים השונים.**
5. רשימה של הקבצים שנוצרו ע"י הסטודנט, עם הסבר קצר (לרוב לא יותר משורה או שתיים) לגבי תפקיד הקובץ.
6. מבני נתונים עיקריים ותפקידיהם.
7. אלגוריתמים הראויים לציון.
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. תכתבו ב־README כל דבר שרצוי שהבודק ידע כשהוא בודק את התרגיל.

אופן ההגשה:

הקובץ להגשה: יש לדחוס כל קובץ הקשור לתרגיל, למעט מה שיצוין להלן, לקובץ ששמו exN_name.zip, כאשר N הוא מספר התרגיל ו־name הוא השם המלא. במקרה של הגשה בזוג, שם הקובץ יהיה לפי התבנית exN_name1_name2.zip, עם שמות המגישים בהתאמה (ללא רווחים; גם בשמות עצמם יש להחליף רווחים בקו תחתי או להצמיד את שני חלקי השם).

לפני דחיסת תיקיית ה־Solution שלכם יש למחוק את הפריטים הבאים:

- תיקיות בשם debug או release (לרוב יש יותר מתיקייה אחת בשם זה, אחת בתיקיית ה־Solution ואחת בתיקיית כל פרויקט).
- תיקייה בשם x64, אם קיימת.
- תיקייה בשם vs. שאמורה להיות בתיקייה עם ה־Solution.

זכרו שגם את הקבצים שנתנו לכם אתם צריכים לצרף, והם צריכים להיות ללא שינוי, כפי שקיבלתם אותם.

ככלל אצבע, אם קובץ ה־zip שוקל יותר ממ"ב אחד או שניים, כנראה שלא מחקתם חלק מהקבצים הבינאריים המוזכרים.

את הקובץ יש להעלות ל־Moodle של הקורס למשימה המתאימה.

הגשה חוזרת: אם מסיבה כלשהי סטודנט מחליט 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. שימו לב לשם הקובץ שאכן יכלול את שמות המגישים.
2. שימו לב שעליכם לשלוח את תיקיית ה־solution כולה, לא רק את קובצי הקוד שיצרתם. עקבו אחרי ההסבר המפורט באתר, במקרה שאתם לא בטוחים איך למצוא את התיקייה. תרגיל שלא יכלול את ה־solution, לא יתקבל וידרוש הגשה חוזרת (עם כללי האיחור הרגילים).

המלצה כללית: אחרי שהכנתם את הקובץ להגשה, העתיקו אותו לתיקייה חדשה, חלצו את הקבצים שבתוכו ובדקו אם אתם מצליחים לפתוח את ה־solution שבתוכו ולקמפל את הקוד. הרבה טעויות של שכחת קבצים יכולות להימנע על ידי בדיקה כזו.

בהצלחה!