מכללת הדסה, החוג למדעי המחשב

תכנות מונחה עצמים ופיתוח משחקים

סמסטר ב', תשפ"ב

תרגיל 4

תאריך אחרון להגשה:

הנביאים - יום א', 08/05/22, בשעה 23:59

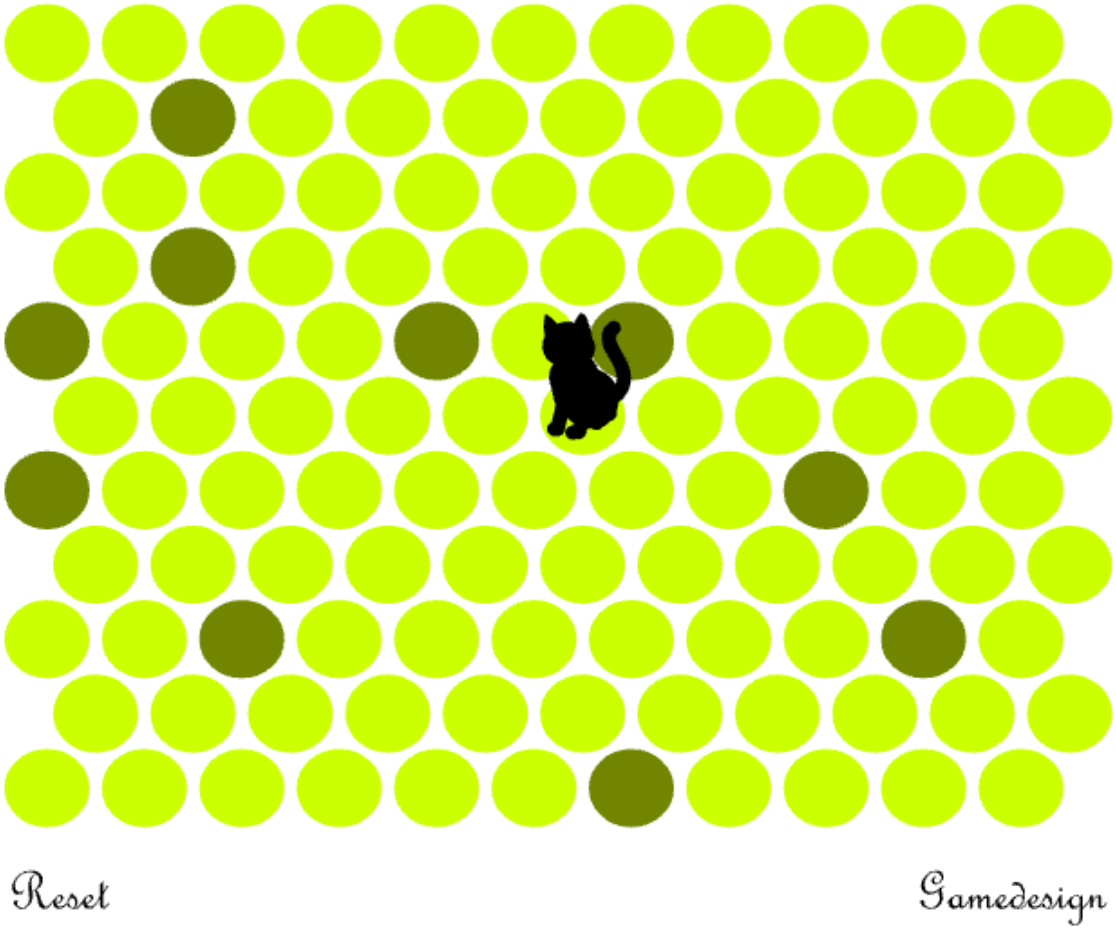
שטראוס-גברים - יום שלישי, 10/05/22, בשעה 23:59

שטראוס-נשים - מוצאי שבת, 07/05/22, בשעה 23:59

מטרת התרגיל:

שימוש במבני נתונים, איטרטורים ואלגוריתמים בסגנון STL וריענון השימוש ב־SFML.

תיאור כללי:



בתרגיל זה נממש את המשחק Circle the Cat.

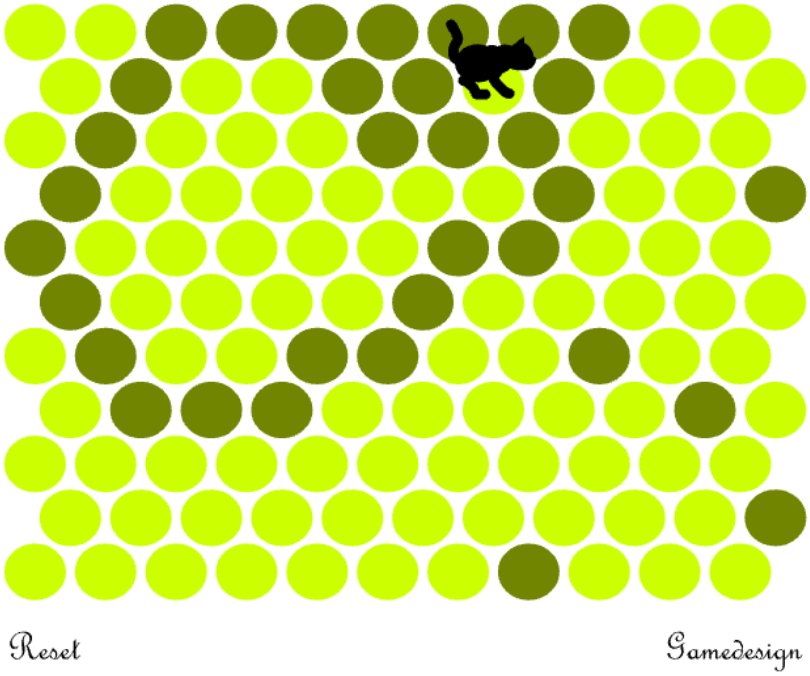
ניתן למצוא הדגמה של המשחק כאן: https://www.crazygames.com/game/circle-the-cat

תיאור המשחק:

משחק זה הוא מז'אנר משחקי הפאזל, בהם מוצג לשחקן לוח עם "חידה" שהוא צריך לפתור לפי כללי המשחק. לשחקן מוצג לוח אריחים (-עיגולים) עם "אריחים חסומים" (-העיגולים הכהים) ו"אריחים ריקים" (-העיגולים הבהירים). במשחק ישנו חתול, שיכול ללכת על העיגולים הריקים בלבד. השלב מתחיל עם מספר עיגולים שכבר מלאים, כשהחתול עומד במרכז על אריח ריק.

מטרת המשחק:

על השחקן להקיף את החתול מכל צדדיו, בעזרת "מילוי האריחים", כך שלא יהיה לו לאן ללכת:



כללי המשחק:

זהו משחק תורות של שחקן מול מחשב. השחקן בתור שלו בוחר אריח אחד שהוא מעוניין לחסום (על ידי לחיצת עכבר על העיגול הרצוי). המחשב בתור שלו מזיז את החתול על מנת לנסות לברוח מחסימות, ועל מנת לברוח מהלוח. אם החתול הגיע לאחד מצדדי הלוח, החתול למעשה ברח מהלוח וניצח במשחק. אם השחקן הצליח

להקיף את החתול מכל צדדיו, כאמור, השחקן ניצח. הערה: על מנת לנצח ולעבור שלב, זה לא מספיק לחסום את החתול במרחב מסוים, כי הוא עדיין יכול "לטייל" בתוך אותו מרחב. צריך לחסום את החתול לגמרי.

פירוט הדרישות:

כשאנחנו מממשים את המשחק, אנחנו לא צריכים "לפתור אותו", זה תפקיד השחקן. אנחנו נדרשים לממש את כללי המשחק: לדאוג "למלא" עיגולים שנלחצו, ולהזיז את חתול למקום "הטוב ביותר" מבחינתו. כלומר נצטרך להחליט להיכן כדאי לחתול ללכת, ולטפל מבחינה אלגוריתמית בבעיה זו (זיהוי המקום "שפחות חסום" על הלוח או לפחות זיהוי דרך שעדיין פתוחה וללכת לכיוון הזה). כמו כן, נצטרך להחליט מה מבנה הנתונים (או מבני הנתונים) שיעזרו לנו לאחסון האריחים ולמימוש האלגוריתם של החתול במשחק.

שימו לב: שהאלגוריתם של בריחה מהלוח יהיה שונה מהאלגוריתם של בריחה בתוך מרחב חסום. הראשון מורכב יותר, ותשקיעו בו.

5 נקודות מהתרגיל (לא בונוס!), יוקצו לטובת הנראות של התרגיל. כלומר משחק שנראה מצויין יקבל את כל ה־5 נקודות. משחק "פשוט" "יתומחר" בהתאם.

פרטי מימוש:

1. אתם נדרשים לממש משחק שנעים לשחק בו, שהכל הולך חלק, ושדברים לא "נתקעים".
2. במשחק שלכם יהיו שלושה שלבים (או יותר) בגודל אחיד של 11X11 אריחים (בהזחה של כל שורה שניה), כשהלוח שונה בין השלבים ברמת קושי עולה (פרטים בהמשך).
3. כאשר השחקן מסיים שלב בניצחון (החתול נחסם), נציג הודעה על ההצלחה ונעבור לשלב הבא.
4. כאשר השחקן מסיים שלב בכישלון (החתול ברח) נציג הודעה על כישלון ונציג **את אותו שלב שוב**.
5. אין חובה לממש תפריט או מסך "שלבים" או לוח שיאים (כמו שאין במשחק המקורי). רק צריך לממש שה־X בפינה יצא מהמשחק.
6. "מונה לחיצות": חובה להציג את מספר הלחיצות שנעשו עד עתה.
7. לחיצה על אריח מלא, או לחיצה "סתם" בכל מקום שאיננו אריח ריק, לא משפיעה על המשחק, ולא נספרת במספר הלחיצות מהסעיף הקודם.
8. עליכם לממש במשחק יוצר שלבים אוטומטי. כלומר המשחק ייצור את השלבים בעצמו על ידי הגרלת האריחים החסומים. מספר האריחים החסומים בתחילת השלב יהיו בין 14 אריחים (שלב קל) לבין 4 אריחים (שלב קשה). טעינת השלבים צריכה להיעשות מהקל לקשה (-לא חייבים להתחיל דווקא ב־14 אריחים חסומים ולסיים ב־4, אלא כל הטווח הוא לגיטימי). כדאי (אין חובה) שיוצר השלבים, ביצירת השלבים הקשים, ינסה לפזר את האריחים, כך שלא יהיו מרוכזים באותו מקום (כך שיהיה ניתן לנצח בו את החתול).

9. **בונוס:** כפתור Undo. מי שילחץ על כפתור זה, המשחק יחזיר אחורה את "הסיבוב האחרון", כלומר יעלים את הפעולה האחרונה (-האריח שנחסם ישתחרר), יחזיר את החתול אחורה למקום שבו היה, ויוריד את מונה הלחיצות ב־1. לחיצה נוספת תחזיר את המשחק עוד "סיבוב" אחד אחורה. וכן הלאה, בלחיצות נוספות, עד לתחילת המשחק. בונוס זה יזכה ב־4 נקודות. (הערה: מימוש כפתור Undo שמחזיר רק את הפעולה האחרונה, ולא יודע לחזור עוד אחורה בלחיצות נוספות – לא יזכה בניקוד כלל).

הערות למבנה הנתונים:

אפשרות די טבעית למימוש המשחק היא בעזרת גרף ואלגוריתם מתאים לטיול בגרף, כמו BFS או DFS (כדי לזהות את הדרך הקצרה ביותר לאחד מצדדי הלוח), אבל אולי ייתכנו אפשרויות טובות אחרות.

אם תממשו מבנה נתונים כזה, כדאי לבנות אותו יחסית גנרי, לא רק מוכוון למטרה הספציפית שלנו כרגע, אבל צריך גם להיזהר לא לנסות לקחת את הדברים לכיוון גנרי מידי, לפי העיקרון שקיים בחלק מהשיטות של הנדסת תוכנה (למשל, XP, Extreme Programming) המכונה: YAGNI, ראשי תיבות של You Ain't Gonna Need It...

הערות למימוש הגרפי ב־SFML:

כדי לממש את האריחים והחתול, אפשר להשתמש בתמונות שתמצאו ברשת או אפשר לממש בעזרת אובייקטים של מלבנים ועיגולים של SFML (זכרו: 5 נקודות מוקצות לטובת ה"נראות" של התרגיל).

הערות לתיכון התוכנית:

שימו לב שיש בתרגיל למעשה שלושה חלקים:

1. מבנה הנתונים שאנחנו נעזרים בו, האיטרטורים והאלגוריתמים.
2. המימוש של הלוגיקה של המשחק.
3. המימוש הגרפי.

כדאי **ומומלץ מאוד** להפריד בין מימוש מבני הנתונים, מימוש הלוגיקה ומימוש הגרפי, גם אם בסופו של דבר ברור שהכול מתחבר יחד לכדי תוכנה אחת שמפעילה את כל החלקים.

כרגיל, את ההחלטות שתחליטו תתעדו כראוי בקובץ ה־Readme. בפרט יש צורך לתעד את ההחלטות על האלגוריתם/האלגוריתמים להזזת החתול, בחירת מבני הנתונים ואופן השימוש בהם והשימוש בספרייה הסטנדרטית. תכתבו במה בחרתם להשתמש ולמה העדפתם מבנה נתונים או אלגוריתם מסוים על פני האחרים. תתעדו גם איך התרגיל שלכם מתחלק לשלושת החלקים שהוזכרו לעיל (למשל פירוט איזו מחלקה משוייכת לאיזה חלק).

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 ומתחתיה 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_firstname_lastname.zip, כאשר N הוא מספר התרגיל ו־firstname_lastname הוא השם המלא (לדוגמא אלברט איינשטיין יגיש כך את התרגיל הראשון: ex1_albert_einstein.zip). במקרה של הגשה בזוג, שם הקובץ יהיה לפי התבנית exN_firstname1_lastname1_firstname2_lastname2.zip, עם שמות המגישים בהתאמה (ללא רווחים; כלומר, גם בשמות עצמם יש להחליף רווחים בקו תחתי, כפי המודגם לעיל). כמו כן, במקרה של הגשה בזוג, **רק אחד** מהמגישים יגיש את הקובץ ולא שניהם.

לפני דחיסת תיקיית הפרויקט שלכם יש למחוק את הפריטים הבאים:

- תיקייה בשם out, אם קיימת
- תיקייה בשם vs.

שתי התיקיות האלה נמצאות בתיקייה הראשית (זו שאנחנו פותחים בעזרת VS). התיקייה vs. לפעמים מוסתרת, אבל אם תפתחו את קובץ ה־zip שיצרתם, בוודאי תוכלו למצוא אותה ולמחוק אותה.

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

וודאו כי קובץ ה־zip מכיל תיקייה ראשית אחת, ורק בתוכה יהיו כל הקבצים ותתי התיקיות של הפרויקט.

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 הפרוייקט כולה, לא רק את קובצי הקוד שיצרתם. תרגיל שלא יכלול את כל הקבצים הנדרשים, 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 התיקייה הזו ולקמפל את הקוד. הרבה טעויות של שכחת קבצים יכולות להימנע על ידי בדיקה כזו.

<u>בהצלחה!</u>