***כותרת:***

משחק "BomberMan".

===================================================

**פרטי סטודנטים:**

שמות:

* יעקב בונדר, ת"ז: 322336488.
* שלי צ'אוסובסקי, ת"ז: 323998526

**הסבר כללי על המשחק:**

ייצוג דמויות:

רובוט (המשתמש) - '\'.

דלת - 'D'.

שומר - '!'.

גבול – '#'.

אבן – '@'.

פצצה עם רובוט – '%'.

פצצה – מספרים הולכים וקטנים כל תור, מ4 עד 1. בסוף הספירה, ייצוגו 5 כוכביות, בצורת פלוס ממיקום הפצצה. (הכוכבית לא פוגעות / מופיעות על דלת או על קיר).

חוקי משחק תקין

* רובוט אחד בדיוק.
* דלת אחת בדיוק.
* שומר אחד לפחות.

אם הקובץ לא מכיל את החוקים הללו, המשחק לא נחשב תקין, ולכן המשחק לא ייטען.

תורות:

הרובוט (המשתמש) מבצע את התור שלו, ולאחריו כל שאר האבוייקטים (שומרים ופצצה) בתור שלהם.

למשתמש יש מספר אפשרויות לביצוע התור:

* לחיצה על אחד ממקשי החצים:
* חץ למעלה (הרובוט זז למעלה)
* חץ למטה (הרובוט זז למטה)
* חץ ימינה (הרובוט זז ימינה)
* חץ שמאלה (הרובוט זז שמאלה)
* לחיצה על המקש 'b', ואז השחקן מניח פצצה.
* לחיצה על המקש רווח, ואז השחקן מאבד את תורו.

תנועה של האובייקטים:

אם התנועה של אחד מהשחקנים היא לעבר קיר או אבן, התנועה לא תתבצע.

נבדיל בין משתמש לשומר:

* משתמש – לא מאבד את תורו, ועד שהוא לא מקבל הוראה תקינה המשחק לא ממשיך.
* שומר – מאבד את תורו, ולא זז.

השחקן יכול לזוז לתא של שומר, אבל הוא יפגע (וכך גם שומר יכול לזוז לתא של שחקן).

כל שחקן יכול לעלות על פצצה מתקתקת, אך נראה דברים שונים:

* כאשר ישנה פצצה ושחקן ביחד, נסמן את התא ב – '%'.
* כאשר ישנה פצצה ושומר ביחד, נסמן את התא ב – '!' (כלומר בשומר).

הורדת חיים:

שחקן יכול להיפצע ממספר דברים:

* שומר פוגע בו.
* הוא נמצא בטווח פגיעה של הפצצה.

אם שומר נפגע מפצצה, הוא נעלם מהשלב.

אם השחקן נפגע, כל השחקנים (הרובוט והשומרים) חוזרים למקומם המקורי, כפי שהיו בעת יצירת המשחק (אם שומר מת, לא נחזיר אותו וכל הפצצות שלא התפוצצו יעלנו גם כן).

ברגע שיש את הפצצה, הכוכביות נשארות תור אחד, וכאשר הכוכביות מופיעות, כל דמות שהייתה בפנים או נכנסה לכוכביות, נפגעת.

**הערה:** פגיעה מפצצה, הינה הימצאות של השחקן בגבול צורת הפלוס, אשר מתחילה ממיקום הפצצה. (נוכחי, למעלה, למטה, ימינה ומשאלה).

סיום המשחק:

המשחק נגמר במצבים הבאים:

* אין עוד קבצי משחק לטעון.
* ברגע שהשחקן נפסל, ישנם שתי אפשרויות לביצוע:
* אפשרות קניית חיים בעזרת הנקודות שהמתמש צבר (אם יש לו מספיק), ואז טעינת המשחק הבא. (בכדי לבחור במספר החיים, יש לבחור מספר בין 0 אשר מייצג ויתור על קניית חיים נוספים, או מספר הגדול מספיק בכדי שיוכל לקנות חיים).
* ויתור על קניית חיים נוספים, וסיום המשחק (המשחק הבא לא ייטען), ותוצג בפני המתמש נתונים אודות מספר הנקודות אשר צבר + מספר החיים שסיים איתם.

ניקוד:

הרובוט יכול לצבור נקודות באופן הבא:

* שומר שהרג – 5 נקודות.
* סיום שלב – 25 נקודות + 3 \* מספר השומרים הכללי שהיה בתחילת המשחק.

**רשימה של קבצים שיצרנו**

Board.h – מכילה את הגדרת המחלקה Board.

Board.cpp – מכילה את הבנאים והפונקציות השונות של מחלקת Board (מימוש המחלקה).

Bomb.h- מכילה את הגדרת המחלקה Bomb.

Bomb.cpp - מכילה את הבנאים והפונקציות השונות של מחלקת Bomb (מימוש המחלקה).

Controller.h- מכילה את הגדרת המחלקה Controller.

Controller.cpp- מכילה את הבנאים והפונקציות השונות של מחלקת Controller (מימוש המחלקה).

Guard.h- מכילה את הגדרת המחלקה Guard.

Guard.cpp- מכילה את הבנאים והפונקציות השונות של מחלקת Guard (מימוש המחלקה).

Location.h- מכילה את הגדרת struct Location.

Location.cpp- מכילה את מימוש הפונקציות של struct Location. פונקציות השוואה בין אינדקסים של שורות ופונקציות השוואה נוספות אשר יעזרו לנו בתזוזה של השחקן.

Player.h- מכילה את הגדרת המחלקה Player.

Player.cpp- מכילה את הבנאים והפונקציות השונות של מחלקת Player (מימוש המחלקה).

Playlist.txt- קובץ בו נכניס את כל השמות משחקים כפי שיפורט בהמשך.

**מבני נתונים עיקריים ותפקידם:**

Vecor - בהם שמרנו את החיילים, כל שורה בלוח (בעצם יצרנו וקטור המכיל וקטורים וזה הלוח בשלמותו) פצצות ואבנים (בעצם כל אובייקט שיש או יכול להיות לנו כמה פריטים ממנו שמרנו במנה נתונים זה).

Location – מחלקה המאחסנת מיקום בלוח. מאפשרת לבצע השוואה בין נקודות, בדיקת נקודות צמודות ועוד פונקציות המקלות על השוואה בין תאים.

**אלגוריתמים הראויים לציון:**

תזוזת החיילים:

במשחק יצרנו אלגוריתם האחראי לתזוזת החיילים, האלגוריתם עובד בדרך הבאה:

* אם השומר צמוד לרובוט, כלומר: מעליו, מתחתיו, מימינו או משמאלו, אזי הוא נכנס לתא של הרובוט.
* אם השומר במיקום אלכסוני לרובוט, אזי הוא לא זז (מונע מהרובוט לנוע לשני כיוונים).
* השומר מחשב את המרחק שלו מהרובוט בשני הצירים, והולך לכיוון הרובוט בציר הרחוק יותר. (המטרה העיקרית היא להגיע לרובוט ולחסום אותו, לכן נרצה לצמצם את הפער בינו לבין הרובוט, וכך להגיע למצב 2).
* במידה והתא שהוא עומד להיכנס אליו לא חוקי, אזי השומר נשאר במקום.

**תיכון (design**)

בקוד המשחק יש לנו מחלקה בשם Controller האחראית לפתיחת PlayList.txt, הוצאה של המשחקים מתוכו והטענת השלבים הבאים על פי סדר שנקבע ב - PlayList.txt.

בנוסף המחלקה אחראי להזזה של כל השחקנים ושליטה על כל תהליך משחק.

בנוסף יש לנו מחלקה בשם Boardהמכילה את האובייקט הלוח, האחראית ליצירה של הלוח, שמירתו ועדכונו בהתאם להתפתחות המשחק (כאן גם נכנס האם הפעולות המבוצעות חוקיות על פי כללי המשחק).

בנוסף יש לנו מחלקה בשם Player, המייצגת את האובייקט רובוט. היא האחראית לתזוזת הרובוט, שמירה, ועדכון המיקום של הרובוט בהתאם לצורך.

בנוסף יש לנו מחלקה בשם Bomb המכילה את האובייקט פצצה, האחראית ליצירה של פצצה בהתאם לבקשה של המשתמש, עדכון הטיימר של הפצצה ובעת פיצוץ עדכון ל - Controller.

בנוסף יש לנו מחלקה בשם Guard המייצגת אובייקט שומר. מחלקה זו אחראית לשמירת מיקום של השומר, הזזה של החייל לפי האלגוריתם שתיארנו ועדכון החייל בהתאם למצבו (אם נפגע או לא).

**באגים ידועים:**

**הערות נוספות:**

טעינת שלבי משחק:

מתבצעת על ידי Playlist.txt. בקובץ זה יש את כלל שמות שלבי המשחק.

אם ברצוננו להוסיף משחק חדש, נוסיף בסוף הרשימה של Playlist.txt ונעדכן ב - CmakeList.txt את השם שלו (יש כמובן גם להוסיף את הקובץ עצמו לתיקיית RESOURCES).

**הערה:** שמות השלבים יכולים להיקרא בכל דרך שהיא (חשוב לא לשים רווח בשם הקובץ, או שהקובץ לא יעבוד, ולא יפתח).

איך ניתן להוסיף משחק חדש?

יש לכתוב את השורה הבאה ב - CmakeList.txt:

configure\_file ("LevelList.txt" ${CMAKE\_BINARY\_DIR} COPYONLY)

גבולות המשחק- הגבולות ניתנות לסימון על ידי התו '#', אך לא בהכרח חשוב שהם יהיו בקובץ המשחק על מנת שהוא יפעל כהלכה.

כאשר אין גבולות ויזואליים, עדיין המשחק חסום על ידי גודל הקובץ של המשחק, ושום שחקן לא יכול לצאת מהגבולות הללו (וכך גם סימון הכוכביות של פיצוץ הפצצה).