**Bomberman**

**תיאור:**

בתרגיל זה כתבנו את משחק הבומברמן שכתבנו בתרגיל 2, אך הפעם עם גרפיקה של ספריית SFML ומשחק שמתנהל בזמן אמת.

במשחק יש כמה שלבים והשחקן עובר שלב כאשר הוא מגיע לדלת היציאה. אם נפסל ונגמרו לו החיים יעבור למסך .GameOver

התוכנית תומכת באפשרות לעשות שלב בלי הגבלת פצצות ובלי הגבלת זמן. כמו כן, יש כפתור restart שמאתחל את השלב, במקרה ולשחקן אין יכולת התקדמות (נגמרו הפצצות וכו'..). יתר-על-כן, באפשרות המשתמש לבחור האם השומרים בשלב יהיו חכמים או טיפשים או בחירה רנדומלית שלהם.

המשחק מנוהל ע"י GameController  
בהתחלה מוצג המסך הראשי – MainScreen הכולל אפשרות התחלת משחק או יציאה.  
מסך המשחק – LevelScreen מחזיק את כל המידע הקשור למשחק:  
הוא כולל GameMenu שהוא התפריט עם המידע על המשחק (מספר שלב, מספר פצצות, ניקוד וכו')  
כמו כן, הוא כולל LevelMap שמחזיק את מבנה הנתונים של הדמויות CharacterList וכן מטריצת קרובים (מאופיינת ע"י (CharMatrix עם רשימות של מצביעים לדמויות (יפורט בהמשך)  
אם המשתמש נפסל – אז הוא עובר למסך GameOverScreen  
אם המשתמש ניצח – אז הוא עובר למסך WinScreen

**הסבר למשתמש:**במסך הראשי יש אפשרות להתחיל משחק או לחלופין לסיים אותו. הבחירה הינה ע"י העכבר.  
במסך המשחק התנועה היא ע"י 4 מקשי הכיוון וזריקת פצצות ע"י המקש b.  
כמו כן, כפתור ה-restart כמו יתר הכפתורים בכל המסכים מופעל באמצעות העכבר.  
מסך הפסילה ומסך הניצחון הם עם כפתורים שפועלים ע"י העכבר בדומה למסך הראשי.  
בכל המסכים ניתן לצאת ע"י מקש Escape.

**פורמט קובץ השלב:**

בתיקיית ההרצה יש קובץ board.bomb המכיל את המידע על כל השלבים.

הקובץ בנוי באופן הבא:

ההפרדה בין שלבים היא ע"י שורה ריקה אחת בלבד בקובץ.

כל שלב מכיל את הנתונים הבאים לפי סדר:  
שורה ראשונה:  
מספר פצצות זמינות. אם הערך 1- אז יש ללא הגבלה.  
זמן ההגבלה של השלב בשניות. אם הערך 1- אז השלב ללא הגבלת זמן.  
סוג השומרים בשלב: יתכנו 3 ערכים:

|  |  |
| --- | --- |
| ערך | משמעות |
| 0 | כל השומרים טיפשים |
| 1 | כל השומרים חכמים |
| -1 | "יותר מזל משכל" - תתבצע בחירה רנדומלית של סוגי השומרים. |

שורה שנייה:  
מספר השורות ומספר העמודות של המפה.  
יתר השורות:  
מפת השלב. התווים הבאים מסמלים את הדמויות במפה:

|  |  |
| --- | --- |
| תו | משמעות |
| ! | שומר |
| @ | סלע |
| / | שחקן/רובוט |
| D | דלת יציאה |
| + | מתנה גלויה |
| & | מתנה מתחת לסלע |
| # | קיר |
| רווח | אין דמות בתא |

הערה: סוג המתנה מוגרל באופן אקראי תמיד ביצירת השלב (נשים לב שאם השלב ללא הגבלת פצצות אזי לא יהיו מתנות מהסוג הזה. אותו דבר לגבי הזמן)

**רשימת קבצים:**  
נחלק בין קבצים הקשורים לממשק המשתמש הגרפי (GUI) לבין קבצים הקשורים ספציפית למשחק:

GUI:

|  |  |
| --- | --- |
| שם הקובץ | הסבר קצר |
| View | מחלקה המייצגת אובייקט שמוצג על המסך. כוללת מיקום, רוחב, גובה, רקע (תמונה או צבע), גבול (צבע וגודל) כוללת אירועים: לחיצה, כניסה ויציאה עם העכבר. פונקציה לציור, קבלת מלבן חוסם, מצב תצוגה/הסתרה, מצב אפשור/ביטול אירועים.  לא ניתן ליצור מופע של המחלקה באופן ישיר אלא ע"י ירושה בלבד. |
| TextView | מחלקה המייצגת טקסט. ניתן לשנות את הפונט, הטקסט וצבעו, ואת כיוון היישור שלו (ימין, מרכז או שמאל) |
| AnimationView | אנימציה. יורש View. מכיל וקטור של ImageTexture וטיימר, ובכך מצייר אנימציה. כולל אופציה לשינוי תדירות עדכון התמונות. |
| ViewGroup | מחלקה המכילה וקטור של אובייקטים (View).  מטרת המחלקה לגרום לכך שכל אירוע שמתקבל יגיע לאובייקטים המוכלים ב- ViewGroup  כמו כן, גם מצב התצוגה והאפשור מועבר לילדים. כלומר, לאובייקטים המוכלים בו.  לא ניתן ליצור מופע של המחלקה באופן ישיר אלא ע"י ירושה בלבד. |
| VerticalLayout | שכלול של ViewGroup שמסדר את האובייקטים בצורה אנכית אחד אחרי השני. גודל האובייקטים מחושבים באופן יחסי בהתאם לממדי ה- VerticalLayout |
| HorizonalLayout | שכלול של ViewGroup שמסדר את האובייקטים בצורה אופקית אחד אחרי השני. גודל האובייקטים מחושבים באופן יחסי בהתאם לממדי ה- HorizonalLayout |
| RelativeLayout | שכלול של ViewGroup שמסדר את האובייקטים באופן מותאם אישית. גודל האובייקטים מחושבים באופן יחסי בהתאם לממדי ה- RelativeLayout |
| ImageView | אובייקט המייצג תמונה |
| imageButton | שכלול של VerticalLayout המכיל ImageView ו-TextView |
| ImageTexture | מכיל מידע על תמונה או צבע ומצייר אותה בהתאם למלבן חוסם שניתן לו |
| Border | גבול – מכיל צבע ועובי הגבול ומצייר אותו בהתאם למלבן חוסם שניתן לו. |
| RectUtils | פונקציות עזר עבור מלבנים |
| IHandler | מנגנון אירועים גנרי Template. מכיל וקטור של פונקציות שמאזינות לאירוע מסוים. כולל פונקציות הוספת האזנה וקריאה לאירוע. |
| StopWatch | שעון עצר. מחזיק שעון עם פונקציית סיום ושעון עם פונקציה שמופעלת כל מרווח זמן.  פונקציות התחלת שעון וסיום שעון. ובדיקה של השעון. משתנה בוליאני האם השעון פעיל. |
| Timer | טיימר. מחזיק שעון עם פונקציה שמופעלת כל מרווח זמן.  פונקציות התחלת שעון וסיום שעון. ובדיקה של השעון. משתנה בוליאני האם השעון פעיל. |

משחק:

|  |  |
| --- | --- |
| שם הקובץ | הסבר קצר |
| GameController | מנהל המשחק. מריץ את מסכי המשחק ושולט על מהלך המשחק - אירועים, התנגשויות וציורים על המסך. |
| MainScreen | מסך ראשי בפתיחת המשחק. מכיל שני כפתורים- התחלת המשחק ויציאה. |
| WinScreen | מסך ניצחון בעת סיום כל השלבים. מציג את הנקודות. מכיל כפתור חזרה למסך ראשי. |
| GameOverScreen | מסך פסילה כאשר נגמרו כל החיים של השחקן. מכיל כפתור חזרה למסך ראשי. |
| LevelScreen | מסך הרצת השלב. מחזיק את תפריט המשחק ומפת השלב. שומר משתנים בוליאניים שאומרים אם יש שימוש בשעון והגבלת פצצות בשלב. |
| GameMenu | תפריט המשחק מופיע בחלק העליון של מסך השלב. מציג את הנתונים עבור השלב. מכיל כפתור restart שמאתחל מחדש את השלב אך גם מוריד חיים. |
| LevelMap | מפת השלב. מחזיק את המטריצה ואת רשימת הדמויות של כל שלב. מחזיק מצביע לשחקן (מטעמי יעילות) ומספר השומרים.  פונקציית טעינת מיקומי הדמויות מחדש בעת פסילה. |
| CharacterList | מבנה נתונים מסוג רשימה מקושרת המחזיקה את רשימת הדמויות. הרשימה תמיד נשמרת ממוינת לפי עדיפות ציור |
| ScoreView | אובייקט שמציג את הניקוד בתפריט הנתונים במסך השלב.  מתעדכן בעת הריגת שומר, סיום שלב וקבלת מתנה מסוג ניקוד. |
| BombsView | אובייקט שמציג את מספר הפצצות שעומדות לרשות השחקן. נמצא בתפריט הנתונים במסך השלב. מתעדכן בעת קבלת מתנה מסוג פצצה או בעת הטלת פצצה. מוסתר כאשר השלב ללא הגבלת פצצות. |
| LifeView | אובייקט שמציג את מספר החיים של השחקן. נמצא בתפריט הנתונים במסך השלב. מציג מספר לבבות כמספר החיים. מתעדכן בעת פסילה ובעת קבלת מתנה מסוג חיים. |
| TimeLeftView | אובייקט שמציג את הזמן שנותר עד לפסילה בשלב. נמצא בתפריט הנתונים במסך השלב. מחזיק שעון עצר ופונקציות איתחול וסיום. |
| NumLevelView | אובייקט שמציג את מספר השלב. נמצא בתפריט הנתונים במסך השלב. |
| Character | אובייקט דמות. מחלקה מופשטת. מחזיקה גישה לLevelScreen-, מיקום במטריצה ומיקום וגודל יחסיים במפת המשחק וכן פונקציות set/get בהתאם.  פונקציות לא ממומשות (pure virtual) של התנגשויות בדמויות אחרות. פונקציות לטיפול בהתנגשויות עם דמויות אחרות במפה. |
| DynamicCharacter | דמות זזה. מחלקה מופשטת. מחזיקה שעון ומהירות כדי לעדכן שינוי מיקום, וכן את כיוון התזוזה (4 כיוונים)  פונקציות תזוזה ובדיקה האם ניתן לזוז והאם יש התנגשות – וזאת בהתאם עם אלו דמויות ניתן להתנגש או לחלופין, לעבור דרכן.  פונקציית play לא ממומשת. |
| StaticCharacter | דמות סטטית. |
| Player | השחקן. יורש דמות זזה.  פונקציות האזנה לאירועים. (שימוש במקשים)  פונקציות האזנה לסיום שלב ולפסילה (events ש-game controller ישתמש בו)  מימוש פונקציות התנגשויות. |
| Guard | שומר. יורש דמות זזה. מחזיק enum של שומר חכם/ טיפש.  פונקציות הגרלה של סוג השומרים.  מימוש פונקציות התנגשות. |
| SmartGuard | שומר חכם. יורש שומר. התזוזה שלו פועלת ע"פ אלגוריתם BFS ויש לו פונקציות בהתאם. מחזיק טיימר שמפעיל את האלגוריתם כל מרווח זמן מסוים. |
| StupidGuard | שומר טיפש. יורש שומר. התזוזה שלו רנדומלית, כלומר הולך עד שהוא נתקע ואז משנה כיוון. |
| Bomb | פצצה. יורש דמות סטטית. מחזיקה שעון עצר ומפעילה פונקציית פיצוץ כעבור 4 שניות. יוצרת 5 פצצות מתפוצצות (במיקומה ו4- כיוונים.) |
| BlowingUpBomb | פצצה מתפוצצת. יורש דמות סטטית. מחזיקה שעון עצר עד למחיקה עצמית. |
| Gift | מתנה. יורש דמות סטטית. מחזיקה enumשל חמישה סוגי מתנות: ניקוד, חיים, פצצות, מהירות ותוספת זמן. פונקציות הגרלה של הסוגים עם אפשרות סינון (כי למשל, אם אין הגבלת פצצות אז לא צריך מתנה מסוג פצצה). מימוש פונקציית התנגשות עם השחקן. |
| Wall | קיר. יורש דמות סטטית. |
| Rock | סלע. יורש דמות סטטית. |
| ExitDoor | דלת יציאה. יורש דמות סטטית. |
| Matrix | מטריצה גנרית. template. בשימוש בקריאת מפת השלב וכן ב-BFS |
| Utilities | Namespace הכולל פונקציות עזר לשימוש הקונטרולר והדמויות. |
| Cell | מחלקת עזר המחזיקה תא: מספר שורה ומספר עמודה. |
| LevelFileParser | קורא מידע מקובץ השלבים board.bomb |
| Resources | Namespace הכולל תמונות, אנימציות, חוצצים של סאונד ופונטים. הכל גלובלי כדי לחסוך בזיכרון. |

**עץ הירושה:**

מצורף בתיקייה הנוכחית תחת class\_diagram.png

**מבני נתונים עיקריים:**

* רשימת הדמויות שמורה ברשימה מקושרת הממוינים לפי "עדיפות ציור".  
  הוספת דמות (בצורה ממוינת) והסרת דמות היא ב- O(n) כאשר n הוא מספר הדמויות. לשם כך נעזרנו ב-std::list
* ישנם 2 "תורי בקשות" לניהול הדמויות:  
  1) תור להוספת דמויות.  
  2) תור להסרת דמויות.  
  הסיבה לכך נובעת כי אנו רוצים לשמור על הסדר של הדמויות לקבלת מהלך תקין של המשחק. לכן, באופן הזה אנו מבצעים את ההוספה ואת ההסרה בזמן "נח". כמו כן, זה מוסיף יכולות לא טריוויאליות לניהול המשחק, כגון: דמות יכולה להרוג את עצמה וכו'.  
  המימוש בוצע בעזרת std::queue
* מטריצה של לוח המשחק שכל אחד מתאיה היא רשימה מקושרת של מצביעים לדמויות.  
  השימוש בה נעשה בבדיקת התנגשויות (לכל דמות, מעבר על 9 תאים סמוכים ובדיקת חפיפת מלבנים – זמן חישוב קבוע בממוצע!) ובחישוב מסלול לשומרים החכמים (כממדי המטריצה)  
  המימוש בוצע ע"י המטריצה הגנרית Matrix<T> שבנינו וע"י std::list

**אלגוריתמים הראויים לציון:**

* שומרים חכמים משתמשים באלגוריתם BFS לחישוב מסלול הקצר ביותר אל הרובוט. לכל שומר שכזה, בכל שנייה, מחושב מסלול מחדש (אם קיים כזה כמובן) הנשמר בווקטור של תאים, והשומר הולך על פיו לעבר הרובוט. אחרת, הוא הולך באופן רנדומלי עד שנתקבע בדומה לשומר הטיפשון.

**הערות:**

* בכל המחלקות ישנה פונקציה toString() שמטרתה לצורכי דיבאגינג בלבד.   
  כמו כן, חלק מהמחלקות זורקות חריגות מסוג string ששימש גם לכך.
* בחלק מהמחלקות היורשות הוספנו בקובץ הכותרת הצהרות מהצורה using BaseClass::func בחלק הפרטי, זאת בכדי למנוע גישה לפונקציות הללו באופן ציבורי (לא עשינו ירושה פרטית, כי חלק אחר של הפונקציות בשימוש)
* באופן חריג, השתמשנו ב-RTTI ב-typeid בבדיקת אזור פנוי עבור הדמויות הזזות.  
  נסביר: במחלקה DynamicCharacter יש וקטור מסוג string שמכיל רשימת מחלקות של דמויות אשר הדמות הזזה אינה יכולה לעבור דרכם. כל דמות זזה היורשת את המחלקה הזאת יכולה להוסיף רשימת דמויות כאלה.  
  במקרה שלנו: הדמויות הזזות הם רובוט ו-2 סוגי השומרים. לכן, בשומר חכם למשל יצהיר באתחול שלו שהוא אינו יכול לעבור פצצה מתפוצצת – כלומר תהיה סוג של התחמקות מהם, ואז כל פעם שתהיה בדיקת תזוזה למיקום מסוים ייבדק האם לא נמצאת שם פצצה מתפוצצת (אלא אם כן הוא כבר בפנים ואז מאוחר מידי..). זאת בניגוד לשומר טיפש שלא יבדוק זאת כלל.  
  הסיבות שהשתמשו בזה הינם:  
  - זוהי דרך גנרית לבדיקת יכולת תזוזה עבור כל דמות זזה – כלומר, זה חוסך מימוש של הבדיקה בכל מחלקה יורשת של דמות זזה.  
  - בדיקת יכולת תזוזה למיקום נתון קשורה רק לדמויות הזזות. ולכן double dispatch לא היה פתרון יפה כל כך, כי אז היה צורך להוסיף את המימוש לכל הדמויות, בפרט הדמויות הסטטיות.
* מפת המשחק מחושבת באחוזים, לכן תתכן מתיחה או כיווץ של המפה אם היחס בין מספר השורות למספר העמודות גדול/קטן, או לחילופין, מספר הפיקסלים בין רוחב המסך לגובהו גדול/קטן.

**מגישים:**  
מתנאל אביוף  
נחום נגה