1. פיתוח משחק "Bombarman".
2. עקיבא משה סודרנסקי: 326382322, אהרון משה שמסיאן: 205779598

**3. הסבר כללי של התרגיל:**

בפרויקט זה נדרשנו לפתח משחק ״בורברמן״ תוך שימוש בגישת תכנות מונחה עצמים (OOP) המשחק יפותח תוך שימוש בירושה, ופולימורפיזם, וניהול אובייקטים באמצעות מחלקות. המטרה היא לפתח משחק שבו שחקן יכול לנוע, להניח פצצות, וליצור אינטראקציה עם רכיבי המשחק השונים (קיר, סלע, דלת, שומר, מתנה ופצצה).

**4. רשימות קבצי המשחק:**

[GeneralManager.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/GeneralManager.h) מנהל ראשי של המשחק, אחראי על ניהול והצגת חלון הפתיחה (WelcomeWindow) וחלון המשחק (GameWindow).

[BasicWindow.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/BasicWindow.h) מחלקת בסיס לניהול חלון משחק, ממנה יורשות המחלקות WelcomeWindow .GameWindo

[WelcomeWindow.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/WelcomeWindow.h) מחלקה האחראית על חלון הפתיחה של המשחק, כולל תצוגת כפתורי ניווט (Start, Help, Exit).

[Button.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/Button.h) מחלקה אחראית על יצירת כפתורים במסך הפתיחה של המשחק (Start,Help,Exit), מחלקה זו מוכלת בתוך Welcome Window

[GameWindow.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/GameWindow.h) מחלקה האחראית על ניהול חלון המשחק, כולל הגדרת גודל החלון וגודל האריחים בתוכו.

[GameInformation.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/GameInformation.h) מחלקה שאחרית לשמירת הנתונים בתחתית חלון המשחק (שלב המשחק, כמות החיים, נקודות וזמן, וכן נתונים כללים שמנהל המשחק צריך לדעת).

[GameInformation.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/GameInformation.h) מחלקה שאחרית להצגת הנתונים בתחתית חלון המשחק (שלב המשחק, כמות החיים, נקודות וזמן).

**מחלקות סטטיות:**

[FontHolder.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/FontHolder.h) אחראית על טעינת גופנים (Fonts) קבועים לשימוש בטקסטים המופיעים ברחבי המשחק.

[Images.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/Images.h) מחלקה סטטית האחראית על טעינת התמונות פעם אחת. מאפשרת קבלת התמונה המתאימה באמצעות פונקציה המקבלת type מוגדר מראש מתוך enum.

**מחלקות עבור האובייקטים המשתתפים במשחק.**

[Participant.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/Participant.h) מחלקת בסיס לכל המשתתפים במשחק. ממנה יורשות מחלקות האב של המשתתפים הסטטיים, כגון קיר, סלע, דלת, מתנות, ושל המשתתפים הדינאמיים, כגון השחקן, שומר חכם ושומר טיפש.

**אובייקטים נחים:**

[StaticParticipant.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/StaticParticipant.h) מחלקת בסיס לכל האובייקטים הסטטיים במשחק, כגון קירות, דלתות, סלעים ומתנות.

[Wall.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/Wall.h) מחלקה המייצגת קירות במשחק, החוסמים תנועה של שחקנים ושומרים.

[Door.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/Door.h) מחלקה המייצגת את הדלת לשלב הבא, שנפתחת רק לאחר עמידה בתנאים מסוימים.

[Rock.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/Rock.h) מחלקה המייצגת סלעים שניתן לפוצץ באמצעות פצצות, וחוסמים תנועה עד להשמדתם.

[Gift.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/Gift.h) מחלקת אבסטרקטית לכל סוגי המתנות במשחק, הנאספות על ידי השחקן ומעניקות בונוסים שונים.

[GiftAddLife.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/GiftAddLife.h) מחלקה המייצגת מתנה שמוסיפה חיים נוספים לשחקן.

[GiftAddTime.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/GiftAddTime.h) מחלקה המייצגת מתנה שמוסיפה זמן נוסף לסיום השלב.

[GiftFreezeGuards.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/GiftFreezeGuards.h) מחלקה המייצגת מתנה המקפיאה את השומרים לזמן מוגבל.

[GiftKillGuard.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/GiftKillGuard.h) מחלקה המייצגת מתנה שמשמידה שומר אחד מיידית.

**אובייקטים נעים:**

[DynamicParticipant.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/DynamicParticipant.h) מחלקה אבסטרקטית לכלל המתתפים שנעים במשחק.

[Guard.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/Guard.h) מחלקת בסיס לשומר במשחק. מחלקה זו מגדירה את ההתנהגות הכללית של שומר, כולל פעולות כמו תנועה ותגובה לשחקן.

[Player.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/Player.h) מחלקה שמייצגת את השחקן במשחק, עם, מיקום ופעולות השחקן, ובנוסף אפשרויות לניהול אינטראקציה עם כלל המשתתפים במשחק.

[DumbGuard.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/DumbGuard.h) מחלקת שומר טיפש, שמבצע פעולות אקראיות ולא מבצע חישוב חכם או תגובות מתקדמות.

[SmartGuard.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/SmartGuard.h) מחלקת שומר חכם, שמגיב לשחקן באופן חכם ומבצע החלטות מבוססות על המיקום של השחקן והסביבה, תוך שימוש באסטרטגיות חכמות יותר.

**אובייקט פצצה:**

[Bomb.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/Bomb.h) מחלקה שמייצגת אובייקט פצצה במשחק. הפצצה אינה זזה, אך אחראית לפיצוץ בחמשת האריחים הסובבים אותה. מכיוון שהיא לא משתתפת בתנועה ישירה, היא יורשת ישירות מ **Participant** (האב הראשי), ולא מ **StaticParticipant** כך שהיא יכולה לבצע את הפיצוץ בצורה עצמאית ועם התנהגות שונה משאר המשתתפים במשחק.

**מחלקות קבצים קבועים:**

[BombConst.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/BombConst.h) קובץ קבועים שמכיל הגדרות קבועות הקשורות לפצצות במשחק, כמו זמן פיצוץ, רדיוס התפשטות, סוגי פצצות או פרמטרים נוספים הקשורים לפיצוצים.

[GameInfoConsts.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/GameInfoConsts.h) קובץ קבועים שמכיל הגדרות קבועות הקשורות למידע כללי על המשחק, כגון מספר רמות, כמות חיים, נקודות, הגדרות ברירת מחדל או פרמטרים שקשורים למבנה הכללי של המשחק.

[GameWindowConsts.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/GameWindowConsts.h) קובץ קבועים שמכיל הגדרות קבועות הקשורות לחלון המשחק, כגון גודל החלון, כותרת, פרמטרים של תצוגה, ושאר ערכים קבועים שמגדירים את הממשק הגרפי.

[ImagesConst.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/ImagesConst.h) קובץ קבועים שמכיל הגדרות קבועות הנוגעות לתמונות ולגרפיקה של המשחק, כולל מיקומים של תמונות, שמות קבצי גרפיקה או פרמטרים אחרים הקשורים להצגת תמונות במשחק.

[MovementConsts.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/MovementConsts.h) קובץ קבועים שמכיל הגדרות קבועות הקשורות לתנועה במשחק, כגון מהירות תנועה, כיוונים אפשריים, ומגבלות על תנועה של שחקנים ושומרים.

[ParticipantType.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/ParticipantType.h) קובץ קבועים המגדיר סוגים שונים של משתתפים במשחק, כמו שחקנים ושומרים, ומספק אפשרות להבחין בין סוגי המשתתפים השונים לצורך טיפול שונה בכל אחד מהם.

[ParticipantsCharId.h](https://github.com/akivasid/oop1_project-YOUR-NAME-HERE/blob/master/include/ParticipantsCharId.h) קובץ קבועים המגדיר מזהים (ID) עבור משתתפים במשחק, כמו שחקן או שומר, ומאפשר זיהוי ייחודי לכל משתתף על ידי מזהה אופי (char) ייחודי לכל אחד.

1. **פירוט מבנה נתונים עיקריים:**

השתמשנו בשלושה וקטורים נפרדים לצורך ניהול האובייקטים השונים במשחק:

1. **ווקטור לאובייקטים נעים (Dynamic):**  
   std::vector<std::unique\_ptr<DynamicParticipant>> m\_dynamic;בו נשמרים כל המשתתפים הדינאמיים, כגון השחקן והשומרים (חכמים וטיפשים), המנהלים תנועה, אינטראקציה ובדיקות התנגשות באמצעות הפונקציות כמו handleCollision , move.
2. **ווקטור לאובייקטים לא נעים (Static):**  
   std::vector<std::unique\_ptr<StaticParticipant>> m\_static;  
   בו נשמרים כל האובייקטים הסטטיים, כגון קירות, דלתות, סלעים ומתנות, אשר אינם נעים אך משפיעים על המשחק (למשל, כחסמים לתנועה.(
3. **ווקטור לפצצות (Bomb):**  
   std::vector<std::unique\_ptr<Bomb>> m\_bombs;  
   הפצצה איננה נחשבת לאובייקט נעים במובן הקלאסי, אך היא גם אינה סטטית – מאחר והיא אחראית על ביצוע פעולת הפיצוץ השפעתה מתפרסת על חמישה אריחים סביב. לכן, נדרשנו לנהל אותה בווקטור נפרד כדי לשמר את ההגיון של חלוקת האובייקטים, ועדיין לאפשר שימוש בפולימורפיזם הנדרש במימוש התרגיל.

במהלך ריצת המשחק, בהתבסס על הפולימורפיזם, כל אחת מהפעולות (כמו בדיקת התנגשות או הזזה) מועברת כארגומנט לאובייקט המתאים מהווקטור המתאים – וכך נשמרת ההפרדה והארגון הלוגי של הקוד, תוך שמירה על ניהול זיכרון חכם על ידי שימוש בווקטור unique\_ptr.

1. ווקטור לשמירת התמונות:

במחלקה Images ישנו ווקטור מסוג Texture המכיל את כל התמונות של סוגי האובייקטים.

1. **אלגוריתמים חשובים שהשתמשנו בהם.**

**האלגוריתם לתנועה של האובייקט** מתחיל בכך שמבוצעת בדיקה אם האובייקט נמצא על אריח סופי. אם כן, תקלוט כיוון חדש כל אובייקט נע לפי עניינו. לאחר קליטת הכיוון, יש לוודא שהוא חוקי ולא חורג אם חורג הכיוון של האובייקט הוא אפס בX ואפס בY ולא תתבצע תנועה על אובייקט זה. אם הכיוון חוקי, נוודא כי התנועה חוקית מבחינת המשחק כלומר אם האריח החדש יש בו קיר או סלע נעדכן את הכיוון ל0,0 ובניהול נבדוק אם כך קרה ןואם כן לא נאפשר תנועה. אם התנועה חוקית אך יש לעדכן פרטים כלשהם הם יתעדכנו באובייקטGameInformation. אם הכל תקין אובייקט ינוע בכיוון זה אך לא יקפוץ ישירות לאריח הבא, אלא יתבצע תהליך של תנועה הדרגתית בעזרת שעון זמן עד שהאובייקט יגיע לאריח הבא. ואם האובייקט יגיע לאריח סופי חדש, התהליך יחזור על עצמו.

**אלגוריתם לתזוזה שומרים** ראשית, על כל ארבעה שומרים טיפשים שמנו שומר חכם אחד. תנועת השומר החכם רודפת אחרי השחקן לפי כיוון תנועתו. אם השומר נתקל באובייקט חוסם, הוא מבצע תזוזה רנדומלית, מה שמאפשר משחקיות מהנה. תנועת השומר הטיפש, לעומת זאת, היא אקראית לחלוטין – הוא זז לכיוון רנדומלי בכל שלב.

**7. תיכון הפרויקט:**

**מבנה התוכנית ומערך הניהול**

התוכנית מנוהלת באמצעות מספר רכיבים מרכזיים, המופרדים על פי תחומי אחריות מוגדרים:

* **GeneralManager** – אחראי על ניהול הבחירה הראשונית של המשתמש.
  + פועל באמצעות חלון הפתיחה **WelcomeWindow** (יורש של **BasicWindow**).
  + אם המשתמש בוחר להתחיל במשחק (**Play**), נוצר אובייקט **GameManager**.
* **GameManager** – מנהל את מהלך המשחק בפועל.
  + משתמש ב**GameWindow** (היורש גם הוא מ-**BasicWindow**) לצורך התצוגה.
  + שולט על מידע המשחק דרך **GameInformation**.
  + עושה שימוש בוקטורים של **unique\_ptr** כדי לאפשר שימוש בפולימורפיזם ולנהל את אובייקטי המשחק באופן דינמי.

**מערכת הירושה של משתתפי המשחק**

בבסיס המערכת עומד המחלקה **Participant**, אשר מייצגת את כלל האובייקטים במשחק. ממנה נגזרות שתי היררכיות עיקריות:

**אובייקטים דינמיים (DynamicParticipant)**

משתתפים במשחק בעלי תנועה ויכולת אינטראקציה דינמית.

* **Player** – מייצג את דמות השחקן הראשי.
* **Guard** – שומרי המשחק, אשר מתחלקים לשני סוגים:
  + **Smart** – שומרים בעלי התנהגות חכמה.
  + **Dumb** – שומרים בעלי התנהגות פשוטה.

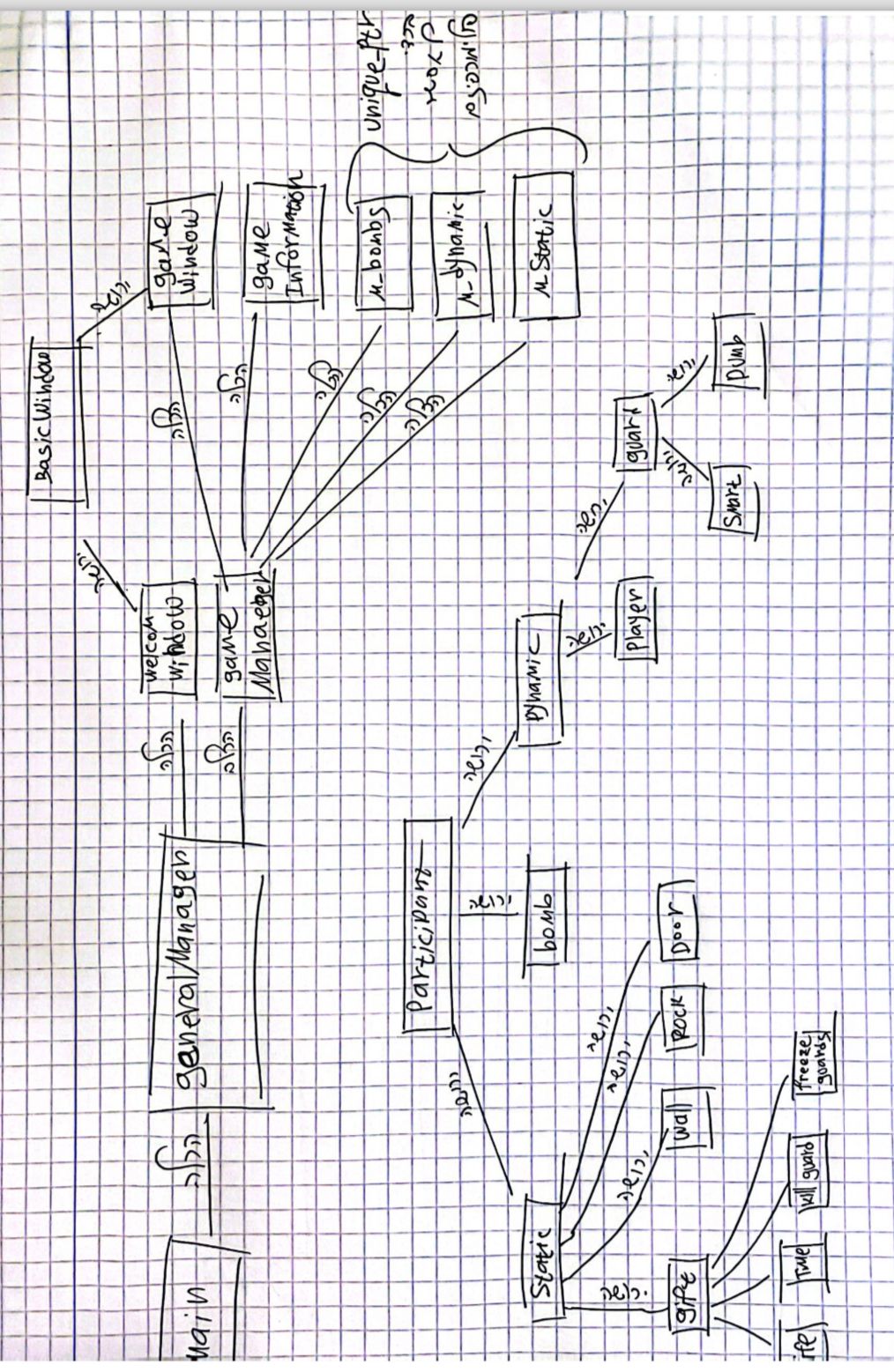
**אובייקטים סטטיים (StaticParticipant)**

אובייקטים נייחים אשר משפיעים על סביבת המשחק.

* **Rock** – סלעים החוסמים מעבר.
* **Wall** – קירות המשחק.
* **Door** – דלתות המעבר.
* **Gift** – מתנות מיוחדות המעניקות יכולות נוספות לשחקן.
  + **GiftLife** – תוספת חיים.
  + **GiftTime** – הארכת זמן המשחק.
  + **GiftKillGuard** – השמדת שומר במשחק.
  + **GiftFreezeGuards** – הקפאת כל השומרים.

בנוסף, קיימת מחלקת **Bomb**, אשר יורשת ישירות מ-**Participant**, ומשמשת כאלמנט אינטראקטיבי במשחק.

מצורפת דיאגרמה



**8. אין באגים ידועים.**

**9. הערות כלליות:**

א. ביצוע התזוזה היא אריחית ומשחקית, כלומר בתנועה אלגנטית ולא קפיצה, כדי שלא מצב שבו אובייקט נע יצטרך לדייק את עצמו בדיוק לפי גודל הפיקסלים.

כלומר מיד בלחיצה אחת לאחד מארבעת הכיוונים אז מתבצע עדכון המיקום של, אך בתזוזה המעשית זה קורה בצורה אלגנטית ומשחקית.

ב. על פי סעיף א, מכיוון שעדכון המיקום מתבצע מיידית לאריח אליו האוביקט צועד, אז ביצוע התנגשות: handleCollision מתבצע ממש לפי ה TopLeft של האובייקיים המתנגשים, ולא בנגיעה בקצוות המיקום של האריחים המתנגשים. ולכן אם שומר נע ממש קצת לכיוון הפצצה הוא ימות, אך אם הוא בכיוון היציאה מהפצצה הוא לא ימות.

ג. חלון המשחק הוא הוא אחד, כלומר לא נסגר ונפתח חלון בין שלב לשלב, אלא החלון הראשון מתאים את עצמו לגודל האריח לפי קובץ התווים של שלב המשחק.