**פרויקט עריכת שלבים בממשק גרפי**

**מגישים:**

1. שלמה זלמן קאהן – 327517066
2. שמעון בריזל – 214552143

**תיאור הפרויקט:**

הפרויקט עוסק בפיתוח מערכת עריכת שלבים באמצעות ממשק משתמש גרפי (GUI). המערכת מאפשרת למשתמשים להוסיף ולשמור שלבים בצורה ויזואלית, תוך שימוש בכלים אינטואיטיביים.

**מבנה הפרויקט:**

הקוד מחולק למספר רכיבים עיקריים, שכל אחד מהם מטפל בפונקציונליות ייחודית:

1. Toolbar.cpp

תיאור כללי:

מודול זה מטפל בניהול תפריט הכלים שבאמצעותו המשתמשים יכולים לבחור פעולות שונות כמו שמירה, ניקוי לוח, הוספת רובוטים ועוד.

פונקציות עיקריות:

* load\_from\_file: טוענת נתוני כלים מקובץ חיצוני (Toolbar.txt), שמאפשר גמישות בניהול הכלים.
* loading\_default\_buttons: יוצר כפתורים ברירת מחדל כמו "שמירה", "ניקוי לוח" ו"שטח פנוי".
* get\_object: מחזיר את האובייקט הנבחר לפי אינדקס.

2. LoadFile.cpp

תיאור כללי:

רכיב זה מנהל את קריאת המידע מקובץ שלבים (Board.txt) ושמירת המידע חזרה לקובץ. הוא גם מטפל בהרחבת הלוח והניקוי שלו.

פונקציות עיקריות:

* fill\_data: קוראת את תוכן קובץ השלב ומכניסה את הנתונים למבנה נתונים (וקטור של מחרוזות).
* update\_data: מעדכנת את הקובץ עם המידע החדש מהלוח.
* check\_if\_robot: בודקת האם רובוט נמצא במיקום מסוים בעת הקריאה מהקובץ עמ לשמור על עיקרון הרובוט (שיהיה אחד בלבד בכל זמן נתון)
* set\_size: משנה את גודל הלוח בהתאם למיקום שהוזן.

3. Controler.cpp

תיאור כללי:

זהו המודול המרכזי שמנהל את התוכנית. הוא מטפל באירועים בזמן אמת, כמו לחיצות עכבר, סגירת חלון ובחירת אובייקטים.

פונקציות עיקריות:

* run: מנהלת את הלולאה הראשית של התוכנית, כולל יצירת חלון ושמירה על רצף עבודה.
* toolbar\_event: מזהה אירועי עכבר בתפריט הכלים ומבצע פעולות מתאימות (שמירה, ניקוי).
* board\_event: מטפל באירועים על הלוח, כמו הוספת רובוטים או מחיקת אובייקטים.
* robot\_control: דואג למיקום הרובוט בלוח, כולל מחיקה והוספה מחדש במיקום חדש.

4. Board.cpp

תיאור כללי:

רכיב זה אחראי על הצגת הממשק הגרפי למשתמש. הוא מנהל את הדמיית הלוח ואת התצוגה של הכלים והאובייקטים.

פונקציות עיקריות:

* update\_window: מעדכנת את המידע הגרפי של האובייקטים והלוח.
* print\_toolbar: מציירת את תפריט הכלים בראש החלון.
* shadow\_object: מציגה את האובייקט שנבחר עם שקיפות לפני מיקום סופי.
* show\_save\_success\_window: מציגה חלון קופץ המאשר שמירה מוצלחת.

**אלגוריתמים מורכבים**

1. ניהול אובייקטים בלוח:

בעת הוספת או מחיקת אובייקט, המערכת משתמשת בפונקציה delete\_object כדי לנקות את המיקום הנוכחי לפני הכנסת אובייקט חדש. זאת מבטיחה שלכל תא בלוח יהיה רק אובייקט אחד, מה שמונע בעיות חפיפה.

2. מיקום רובוט:

בקרת הרובוט מתבצעת על ידי הפונקציה robot\_control, שמוודאת כי רק רובוט אחד נמצא בלוח בכל זמן נתון. האלגוריתם מוחק את הרובוט הקודם ומעדכן את מיקומו החדש.

3. טעינת נתונים מקובץ:

בעת קריאת הקובץ, האלגוריתם fill\_data משתמש בלולאות כדי לקרוא כל שורה ולוודא שהמידע נאגר בצורה תקינה בווקטור. תהליך זה משפר את היכולת לעבוד עם קבצים גדולים.

באגים ידועים:

1. במידה ורוחב (col) החלון קטן מ3 – העמודה הימנית ביותר לא תשמר לקובץ, וכן כל אובייקט בסרגל הכלים יזוז שמאלה לכן ניסיון ללחוץ על הסרגל – ילחץ במיקום ימני מעט יותר מן המצופה. למרות זאת – הנחת אובייקטים במיקומים הנל לא יגרמו לנפילת התוכנית או לזרימתו הרציפה.  
   ניסינו להתמודד עם זה ללא הצלחה, מכיון שרוחב זה נדרש לכפתורי ברירת מחדל בחלון עצמו (כפתורי הסגירה, הגדלת חלון ומזעורו שמוגדרות במערכת ההפעלה).

הערות:

1. דלתות – אפשרי להניח יותר מאחד, מכיוון שלא הוגדר בדרישות התרגיל.
2. בעת שמירה לקובץ (save) – החלון אינו נסגר, וניתן להמשיך לערוך את הלוח.  
   גם זה מהסיבה הנל.