**משחק חוסם שטחים Xonix**

מגישים: יאיר שטיינלוף 314684580  
 גילעד שטרוזמן 318812930

**הסבר כללי:**

בתרגיל זה בנינו את המשחק חוסם שטחים. השחקן בצבע לבן, שומרים ירוקים, שובל הליכה של השחקן אדום, שטח כבוש כחול ושטח שאינו כבוש בצבע שחור. יש מספר שלבים כל שלב קשה יותר מקודמו. מטרת השחקן לכבוש אחוז מסוים מהלוח תוך הימנעות ממגע עם אויבים או חזרה על עקבותיו באמצע תנועה. כל אזור שנסגר ע"י השחקן נכבש ונצבע בהתאם.

**רשימת קבצים שיצרנו:**

* main.cpp - נקודת הכניסה לתוכנית. מפעילה את מנוע המשחק ומטפלת בחריגות כלליות.
* **Engine** - מנוע המשחק: יוצר את חלון המשחק, מפעיל את לולאת המשחק, ומנהל את מצבי המשחק.
* FilesManager - טוען הגדרות כמו רזולוציה, מספר אויבים, מספר חיים וכו' מתוך הקובץ levels.txt
* GameStateManager - מנהל מצבי משחק בעזרת מחסנית של אובייקטים: תפריט, משחק, תצוגת סוף משחק.
* GamePlayState - מצב המשחק הראשי: מנהל את כל מהלך המשחק בפועל כולל שחקן, אויבים, לוח ותוצאה.
* **GameOverState** – מציג מסך הפסד
* **WinState –** מציג מסך ניצחון
* Board - מייצג את לוח המשחק, מטפל בלוגיקת כיבוש שטח, התנגשות בין שחקן לאויבים ועוד.
* **Player** - מייצג את השחקן, כולל חיים, תנועה, אחוז כיבוש וניהול עקבות.
* **Enemy** - אויב שנע באלכסון על הלוח, ומסכן את השחקן.
* **Cell**  - תא בודד בלוח. יכול להיות לא כבוש, כבוש, או עקבות.
* GameObject / MovingObject - מחלקות בסיס לתנועה, מיקום וציור למסך.
* **ScoreBoard** - לוח תוצאות שמציג מידע חיוני: ניקוד, חיים, שלב ואחוז כיבוש.
* **FontManager** - מנהל גופן (arial.ttf) ומוודא טעינה בטוחה ואחידה של גופנים.

**מבני נתונים עיקריים:**

* std::vector<std::vector<Cell>> m\_matrix - מטריצת הלוח (תאים מסוגים שונים).
* std::stack<std::unique\_ptr<GameState>> m\_states - מבנה המחזיק את מצבי המשחק.

**אלגוריתמים הראויים לציון:**

* **אלגוריתם Flood Fill** – מבצע בדיקה האם אזור מוקף על-ידי השחקן בטוח לכיבוש, ואם כן – כובש אותו. אם קיים אויב באזור, הכיבוש מבוטל. מבוצע באמצעות BFS המשתמש בתור כמבנה נתונים. שימוש בlist במקום וקטור כי מבנה נתונים זה יעיל יותר במקרה הנוכחי- צריך רק הכנסות ולא חיפוש. כך נחסך copy של וקטור במקרה של הגעה לגבול. ושימוש בתור לשם אלגוריתם BFS.
* **הפרדת לוגיקה לפי מצבי משחק -** כל מצב משחק תפריט, סוף משחק, ממומש כאובייקט נפרד עם ממשק אחיד.
* **אלגוריתם השמת שומרים-** במקומות רנדומליים שלא יכולים לחזור על עצמם – שימוש בset כמבנה נתונים.

**תיכון:**

* **Engine** - מנהל את לולאת המשחק המרכזית. מכיל GameStateManager, וקורא לפונקציות של States באמצעות שימוש בפולימורפיזם.
* **GameStateManager -** מנהל מעבר בין מצבים שונים.
* GamePlayState - מריץ את המשחק בפועל, כולל כל הרכיבים. מכיל לוח, ובעל מצביע יחודי לשחקן, הוא שולח מצביע לשחקן ללוח, וכך אותו שחקן נשמר במעבר בין רמות, ומחלקה זו אחראית על הריסתו.
* **Board** - שולט בלוגיקת המשחק (תנועה, כיבוש, התנגשות)
* Player / **Enemy** - אובייקטים נעים עם לוגיקת תנועה עצמאית
* **ScoreBoard** - מופרד מציור הלוח ומציג מידע חיוני.
* FilesManager - מייצר ממשק נוח לטעינת קובצי רמות
* GameState- מחלקה וירטואלית שממנה יורשות כל מחלקות הState.