**OOP2 – תרגיל 4:**

**מגישים**:

דור רוזין- 205618267

אילון מזור- 205438369

**תיאור התרגיל**:

בתרגיל זה התבקשנו להביא לידי ביטוי וליישם את המשחק Lights.

מטרת המשחק היא לחבר בין כל הצלעות של הקודקודים כך שהנורה (שנמצאת בקודקוד האמצעי), תוכל להאיר את כל שאר הקודקודים. ברגע שכל הנורות מוארות, השלב הסתיים, והשחקן יועבר לשלב הבא.

הצורה שבה בחרנו את בניית השלבים היא שלושה שלבים הכתובים מראש על ידינו, בפורמט שנסביר בהמשך.

בכל שלב במהלך התוכנית, במידה וקלט מסויים אינו תקין או ששם הקובץ אינו תקין, אנו זורקים חריגה, אשר התוכן שלה נכתב לתוך הקובץ log.txt בתיקייה out.

**תיכון:**

בתוכנית קיימים מספר אובייקטים:

**Board –** אחראי על קריאת הפרטים מהקובץ, הזנתם למבני הנתונים שלנו ובדיקת תקינות הקלט.

**Point –** אחראי על ייצוג של נקודה במשחק. אובייקט זה מחזיק את כל הפרטים הקשורים לנקודה מסוימת- מיקום, מספר צלעות, כיוון צלעות וכדומה.

**Menu –** אחראי על התפריט הראשי ועל המעברים בין שלב לשלב.

**Controller –** אחראי על ניהול המשחק, הרצת לולאות המשחק, בדיקת קיום חוקי המשחק ואלגוריתמים עליהם נרחיב בהמשך.

**Utilities –** מחלקה המחזיקה את קבועי התוכנית.

**קבצים**:

הקבצים שיצרנו הם- level1.txt, level2.txt, level3.txt

כשמם הם, כל קובץ מייצג אחד משלושה שלבים אשר אנו קובעים מראש במשחק.

חשוב לציין – שניתן להגדיל את המשחק לאיזה גודל שרוצים – בתנאי שעומדים בכללי המשחק, תוך התחשבות בגודל החלון. (הגדרנו אותו להיות 500, ניתן לשנות דרך מחלקת Utilities)

**מבני נתונים**:

בתוכנית קיימים מספר מבני נתונים:  
  
הנקודות עצמן נשמרות בוקטור של נקודות (<std::unique\_ptr<Point>> ) תחת ה Controller. (נקודות = מחלקת Point)

כל נקודה מחזיקה וקטור של מצביעים לנקודות (point\*), כלומר – השכנים.  
ובנוסף, כל נקודה מחזיקה וקטור של מלבני SFML.

**אלגוריתמים**:

בתוכנית קיימים מספר אלגוריתמים:

**קריאת נתונים** – קריאת הנתונים מהקבצים מתבצעת בפורמט מסויים אותו הגדרנו בעצמנו-   
חובה על הנתונים להיות מוזנים כך עבור כל נקודה (שורה נפרדת לכל נקודה) :   
א. ערך X.   
ב. ערך Y.  
ג. סדרת מספרים בין 1-6, כאשר כל מספר מחושב להיות הצלע.  
(לדוגמה: עבור המספר 5 – נקבל צלע בזווית שפונה לכיוון 270 מעלות)

ובמקרה הכללי: (עבור x בין 1 ל6 תתקבל הזווית = angle = x \* 60 - 30)

במידה והקובץ יהיה בפורמט לא תקין (תווים שאינם מספרים, מספרים שאינם בטווח, יותר מידי או פחות מידי צלעות -> תיזרק חריגה)

**חישוב שכנים בזמן ריצה –** בתחילת ריצת כל שלב, אנו מחשבים את רשימת השכנים של כל קודקוד, על מנת לבצע ריצה על הגרף. כמו כן, במהלך חישוב השכנים – אנו מוודאים שאין לקדקוד מסוים יותר צלעות משכנים, כך שלמשל לקודקודי המסגרת יש פחות שכנים מקודקודים פנימיים שבתוך הלוח.  
החישוב עצמו מתבצע באופן פשוט, כאשר אנו בודקים את המרחקים בין הקודקודים, וכאשר יש מרחק קבוע ברוחב או בגובה – אנו יודעים שזה שכן. לאחר וידוא מספר השכנים – אנו בודקים שלקדקוד הנתון אין יותר צלעות ממספר השכנים.

**BFS –** לשם חישוב הדלקת/כיבוי הנורות בהתאם לפעולתו האחרונה של השחקן, אנו מבצעים BFS על הקודקודים, כאשר קדקוד המקור הוא הנורה האמצעית, והוא הראשון שנכנס לתור.  
בתחילת האלגוריתם – אנו יוצרים תור, ומכניסים אליו את קודקוד המקור, ולכל השאר מכניסים False. לאחר מכן אנו עוברים סדרתית על התור, ועבור כל קודקוד אנו בודקים את כל הצלעות שלו עם כל השכנים (תוך שיקולי יעילות במידה וקודקוד מסוים כבר בתור וכד'), ובמידה וישנה חפיפה – אנו מוסיפים את הקודקוד שאיתו ישנה חפיפה לתור וממשיכים לעבור על שאר הצלעות.

**באגים ידועים:**

אין.

**הערות:**

לשם פשטות בדיקת המשחק, שלב 1, שהוא הקל ביותר, הוא היחיד אותו צריך לפתור.

שלבים 2 ו-3 יסתיימו כעבור קליק אחד, דאגנו לכך על מנת לסיים את המשחק מהר.