**OOP2 – EX04**

**מגישים:** יצחק חיים הלוי 315706986 , מתן טולידנו 311516272

**הסבר כללי של התרגיל:** בתרגיל זה אנחנו מממשים את המשחק Circle The Cat . המשחק הוא מז'אנר משחקי הפאזל, בהם מוצג לשחקן לוח עם "חידה" שהוא צריך לפתור לפי כללי המשחק. לשחקן מוצג לוח אריחים שמיוצגים בעזרת עיגולים. עיגולים צהובים = אריחים ריקים, עיגולים שחורים = אריחים חסומים. במשחק ישנו חתול, שיכול ללכת על העיגולים הריקים בלבד. השלב מתחיל עם מספר עיגולים שכבר חסומים, כשהחתול עומד במרכז על אריח ריק.

מטרת המשחק היא להקיף את החתול מכל צדדיו, בעזרת מילוי האריחים כך שלא יהיה לו לאן ללכת.

על השחקן בתורו לבחור אריח אחד שהוא מעוניין לחסום בעזרת לחיצת עכבר על העיגול הרצוי. המחשב בתורו מזיז את החתול על מנת לנסות לברוח מהחסימות, ועל מנת לברוח מהלוח. אם החתול הגיע לאחד מצדדי הלוח, החתול למעשה ברח מהלוח וניצח במשחק. אם השחקן הצליח להקיף את החתוך מכל צדדיו, השחקן ניצח. הערה: על מנת לנצח ולעבור שלב, זה לא מספיק לחסום את החתול במרחב מסוים, כי הוא עדיין יכול "לטייל" בתוך אותו מרחב. צריך לחסום את החתול לגמרי.

**תיכון:**

אנחנו משתמשים בספריה של sfml כדי לייצג את המשחק בצורה ויזואלית, כאשר המימוש הגרפי נעשה במחלקת controller .

האריחים מיוצגים במחלקה נפרדת כאשר הם מכילים עיגול מהספרייה sfml , ויש לכל אריח שכנים (מקסימום 6 שכנים), כאשר השכנים שלו גם הם אריחים שנמצאים בסמוך אליו בלוח. במחלקת לוח אנחנו משתמשים בוקטור דו ממדי שישמור לנו את האריחים כך שנוכל לטפל בהם. הסיבה לשימוש בוקטור דו ממדי היא כדי שנוכל לייצג כל אריח גם לפי המיקום שלו במטריצה.

לוח המשחק הוא בגודל 11X11 כאשר בתחילת המשחק (השלב הקל ביותר) מתחיל כאשר החתול באמצע הלוח, ויש 14 אריחים חסומים. המשתמש יכול לראות בצד באיזה שלב הוא נמצא, וכמה לחיצות על אריחים הוא לחץ באותו שלב. במידה והשחקן מנצח, הוא עובר לשלב הבא כאשר רמת הקושי עולה (בסה"כ יש 3 שלבים). במידה והשחקן מפסיד, החתול יחזור למיקום ההתחלתי, וכמות האריחים החסומים תישאר כשהייתה, כלומר רמת השלב תישאר כמו שהייתה.

השלבים נוצרים באופן אוטומטי ללא עזרה של קבצים מבחוץ. בקובץ Resources.h אפשר לשנות את הקבועים NUM\_OF\_LEVELS, EASY\_LEVEL, MEDIUM\_LEVEL, HARD\_LEVEL כדי שיהיה ניתן להוסיף עוד שלבים ולשנות את הרמות של השלבים.

החתול זז בלוח בעזרת אלגוריתם BFS ממומש במחלקת הלוח, כאשר הוא מחפש כל פעם את המסלול הקצר ביותר לעבר הקצוות של הלוח. ההחלטה ללכת על אלגוריתם זה נעשתה בגלל שחשבנו שהוא יהיה הכי מתאים למשחק הזה כי הוא גם יעיל וגם הוא נותן לנו בדיוק את מה שאנחנו צריכים. במידה והחתול נחסם, הוא יזוז באופן רנדומלי עד אשר לא יהיה לו לאן לזוז והשחקן יעבור שלב/ינצח במשחק.

**רשימה של הקבצים שיצרנו (לא הקבצים שקיבלנו עם התרגיל) :**

קבצי header :

Board.h – מייצג את מחלקת הלוח שאחראית על התצוגה שלו.

Cat.h – מייצג את החתול.

Controller.h – אחראי על ניהול האירועים שקורים בזמן אמת על המסך.

Resources.h - תבנית עיצוב Singleton – טעינה וגישה לכל המקורות שבהם אנו עושים שימוש בתכנית כגון טקסטורות, מוזיקה ועוד.

Tile.h – מייצג את האריחים.

קבצי cpp :

Board.cpp – מימוש הלוח עם בניית כל הדמויות תוך תפעול הזזה שלהם על הלוח.

Cat.cpp – מימוש של החתול.

Controller.cpp – מימוש ניהול האירועים שקורים בזמן אמת על המסך.

main.cpp - main עם קריאה לcontroller שיתחיל להריץ את המשחק.

Resources.cpp - מימוש תבנית עיצוב Singleton – טעינה וגישה לכל המקורות שבהם אנו עושים שימוש בתכנית כגון טקסטורות, מוזיקה ועוד.

Tile.cpp – מימוש של האריחים.

**מבני נתונים עיקריים ותפקידיהם:**

וקטור דו ממדי של אריחים כדי שנוכל לייצג אותם בצורה ויזואלית ולהזיז את החתול בהתאם למצב שלהם.

**אלגוריתמים הראויים לציון:**

אלגוריתם BFS . הסתמכנו על הקוד של BFS דרך הקישור:

https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/

**באגים ידועים:**

**הערות אחרות:**