**Digger**

**פרטי הסטודנטים**:

אור שחר 203929047

טל בן שבת 315702712

**הסבר כללי של התרגיל:**

בתרגיל זה נממש את המשחק Digger.

במהלך המשחק, המשתמש שולט בדיגר באמצעות מקשי החיצים, ומטרתו "לאכול" את כל היהלומים בשלב ולעבור לשלב הבא.

בדרך יאלץ להתמודד עם מפלצות שינסו למנוע ממנו לסיים את השלב.

המשחק מסתיים בהצלחה כאשר הדיגר הצליח לעבור את כל השלבים.

**תיכון (design)**:

**Asset\_Manager –** תפקידו לטעון ולנהל את הקבצים החיצוניים שנחוצים למשחק כגון: קבצי שמע ,תמונות ופונטים.

**Menu** – התפריט הראשוני שרץ עוד לפני שהמשחק התחיל. מחזיק מופע של שני כפתורים (התחלת משחק ויציאה).

תפקידו: להתחיל את המשחק או לסגור אותו, על פי בקשת המשתמש.

**Button** – מחלקה של כפתור בודד.

**Board** –מחזיק מופע של כל אחד מהאובייקטים במשחק (או וקטור של מופעים), ועושה את הקישור ביניהם. כמו כן מחזיק Rectangle Shape שמייצג את הלוח שעליו מוצגים כל האובייקטים.

בין תפקידיו: לטעון את האובייקטים מהקובץ, לנהל התנגשויות בין האובייקטים, (באמצעות Double Dispatch).

**Game** – מנהל את המשחק. מנהל את הלולאה הראשית של המשחק, מנהל אירועים של מקלדת(יציאה), אחראי לקריאה לאובייקט של התפריט, להציג הודעות למשתמש וכ'..

**DataBar** – מנהל את הסרגל שמציג את הנתונים במשחק (חיים, ניקוד, זמן, אבנים, שלב).

מחזיק Rectangle Shape משלו שעליו יוצג הסרגל נתונים מעל הלוח.

**Object** – מחלקת הבסיס של כל אובייקט במשחק.

מחלקות היורשות מ Object –

* **Statics** – מחלקת הבסיס של כל אובייקט שאינו זז
* **Movables** - מחלקת הבסיס של כל דמות במשחק

מחלקות היורשות מ Statics –

1. **Wall** – מחלקה של קיר
2. **Stone** – מחלקה של אבן
3. **Diamond** – מחלקה של היהלום
4. **Gift** – מחלקה של מתנה

מחלקות היורשות מ Movables –

1. **Digger** – מחלקה של השחקן (דיגר)
2. **Monsters** – מחלקה של מפלצת שמתפצלת לשני סוגי מפלצות: מפלצת שזזה רנדומלית (**RegularMonster)** ומפלצת חכמה שרודפת אחרי השחקן (**SmartMonster)**.

**פורמט קובץ השלב:**

שם הקובץ: Board.txt

כל השלבים יכתבו בתוך קובץ אחד כאשר בין כל שלב מפריד שורה ריקה אחת. אין הגבלה למספר השלבים שניתן להוסיף.

שורה ראשונה מייצגת את נתוני השלב לפי הסדר הבא:

מספר שורות -> מספר עמודות -> מס' אבנים נותרות לאכילה -> הגבלת זמן לשלב (1- מייצג בלי הגבלה)

ולאחר מכן ציור של מטריצת האובייקטים בשלב.

**רשימת קבצים שנוצרו:**

בנוסף לכל קבצי ה .h וקבצי .cpp של כל המחלקות שהוזכרו לעיל, נוספו הקבצים:

* ClassDiagram.png – דיאגרמת UML של הפרויקט.
* תיקיית resources– כוללת את הקובץ עם השלבים המוכנים, התמונות, הקבצי שמע, וקובץ הפונט הנחוצים למשחק.

**מבני נתונים:**

מבנים מסוג vector (מערך):

ה Game שומר את המערכים הבאים:

1. vector<unique\_ptr<Monsters>> m\_monsters: מערך של מפלצות
2. vector<unique\_ptr<Statics>> m\_statics: מערך של כל החפצים שאינם זזים

ה Asset\_Managerשומר:

map<int, sf::Texture> m\_textures : מפת גיבוב של התמונות הנחוצות לציור המשחק.

map<int, sf::Font> m\_fonts : מפת גיבוב של הפונטים הנחוצים במשחק.

map<int, sf::SoundBuffer> m\_sounds : מפת גיבוב של הסאונדים הנחוצים במשחק.

**אלגוריתמים:**

תזוזת המפלצות:

מפלצת רגילה (Regular Monster) מגרילה מספר בין 0 ל 3(כולל), וכל 2 שניות יוצרת כיוון לפי המספר שהוגרל.

מפלצת חכמה (Smart Monster) לעומת זאת, מקבלת את מיקומו של הדיגר, ומחשבת את המרחק היחסי בינו לבין המפלצת (באמצעות משפט פיתגורס). אם המרחק קטן מ 400 יחידות פיקסלים, המפלצת מתחילה לרדוף אחריו. לאחר מכן המפלצת זזה על ציר ה X או ציר ה Y (בהתאם למיקום הדיגר). במידה והדיגר נפסל (נפגע ע"י מפלצת או נפסל דרך אבנים) או שהשחקן התרחק מספיק, המפלצת מפסיקה לרדוף אחריו.