Space invaders

ראשית חילקתי את התרגיל למרכיבים ובעזרת הבסיס של תרגילים קודמים מימשתי את הממשק הרצוי .
המשחק נקרא מקבצי טקסט ורץ על ידי game flow

השחקן : פאדל עם היכולת לירות ומנגנון הורדת חיים .

ממברים חדשים – משתנה מסוג LONG בשם PAST וכן משתנה בוליאני FIRST מאותחל לTRUE.
מנגנון הורדת חיים – דרך פונקציית hit משזוהה פגיעה , הורדתי חיים ואתחלתי את החייזרים למצבם המקורי .
יריות – דרך timePassed זימנתי את ballCreator ב GAMELEVEL ושלחתי את הפאדל כפרמטר כך שיווצרו קליעים מותאמים בזווית כלפי מעלה ומיקום יחסי לפאדל.
בעזרת PAST אני שומרת תמיד אינדיקציה לפעם האחרונה שירינו – ובמידה והזמן שעבר גדול מ0.35 מתאפשרת יריה .

```
if (keyboard.isPressed("space")) {
    if (first) {
        past = System.currentTimeMillis();
        g.ballsCreator(this);
        first = false;
    } else {
        long now = System.currentTimeMillis();
        if (now - past > 0.35 * 1000) {
            past = System.currentTimeMillis();
            g.ballsCreator(this);
        }
    }
}
```

מגנים : בלוקים אשר נעלמים בהדרגתיות .

בעזרת המערכת הנפלאה של תרגום קבצי טקסט שיצרנו זה מכבר יצרתי 3 סוגי בלוקים כל אחד בעל סיבולת של 3 פגיעות כאשר לאחר פגיעה מתחלפים לתמונה מחוררת של עצמם כל מגן בנוי מקומבינציה של 6 בלוקים משלושת הסוגים שלעיל.
אותם הוספתי למשחק דרך GAMELEVEL , תוך כדי הוספתם למשחק הוספתי אותם גם לרשימת מגינים עליה אני עוברת כדי למצוא את ערך הY הנמוך ביותר אותו אני שומרת כממבר ב GAMELEVEL כך שבעתיד אשלח אותו לאויבים ואדע מתי הם מגיעים לגובה המגינים גם אם המגינים יהרסו עד אז.

חייזרים :

מראה – בעזרת מערכת הקבצים יצרתי סוג בלוק E בגודל 30*40 בעל תמונת חייזר וסיבולת של פגיעה אחת , בקובץ הכללי יצרתי 10 טורים של חמישה חייזרים בכל טור .

יכולות – יצרתי מחלקה חדשה מסוג ENEMY הממשת את COLLIDABLE SPRITE HITNOTIFRE כך שאוכל לרשום את הבלוקים המיועדים להיות אויבים כאויבים.

על מנת לזהות מי אמור להיות אויב יש בבלוק פונקציית בדיקה על פי נתוני מפת ערכים

```
public boolean rUEnemy() {
    return f != null && f.size() == 1;
}
```

פונקציות מיוחדות בגוף האויב:

```
public Enemy(Block b, double sp, double min, double max, double d)
```

בנאי המקבל בלוק , מהירות ,ערך מינימלי ומקסימלי של המסך וכן ערך dt

```
public boolean inRightRangend()
public boolean inRightRangestart()
```

פונקציות הבודקות אם תזוזה היא חוקית ולא עוברת את המסך (בהתאמה תזוזה ימין ושמאל)

```
public void moveLeft()
public void moveRight()
```

פונקציות האחראיות על תזוזה ימין ושמאל על ידי ריפוזיציה של הבלוק של האוייב.

```
public void change()
```

פונקציה המזומנת בזמן של שינוי כיוון ודואגת להורדה בשורה של האויב והגדלת המהירות ב10%

```
public void drawOn
```

מתבצע על ידי זימון פונקציה זו הקיימת בבלוק .

```
public Velocity hit(Ball hitter, Point collisionPoint, Velocity
currentVelocity)
```

רק במידה והכדור הפוגע אינו מוגדר ככדור אויב (משתנה בוליאני בגוף הכדור) נוריד את כמות הפגיעות ונודיע לכל על הפגיעה .

```
public boolean same(Block b)
```

פונקציה הבודקת אם בלוק מסוים זהה לאויב (בדיקת זהות בין הבלוקים).

```
public void shoot(GameLevel gameLevel)
```

פונקציה המסגלת יריה , זימנתי את ballCreator ב GAMELEVEL ושלחתי את האויב הנוכחי כדי שהכדורים יווצרו כלפי מטה במיקום מתאים לאויב והכי הכי חשוב הכדורים יאופיינו ככדור אויב (שינוי משתנה בוליאני אוייב בכדור לTRUE)

```
public void reset()
```

איפוס אויב למיקומו ההתחלתי . בכל פעם שיוצרים אויב אני שומרת כממבר את הבלוק ההתחלתי שלו ואת המהירות ההתחלתית שלו – כאשר פונקציה זו מופעלת אני מאתחלת אותו למצבו ההתחלתי על ידם.

טיפוס זה מנוהל מתוך מחלקה גדולה יותר בשם ENEMIES אשר אחראית לתזוזה האחידה והירייה רנדומלית .

```
private List<Enemy> enemys = new ArrayList<Enemy>();
```

רשימת כל האויבים במשחק.

```
private boolean right;
```

ממבר בוליאני המודיע על כיוון

```
public Enemies(double sp, double mn, double mx, GameLevel gi)
```

בנאי כללי המקבל מהירות ערכי מינימום ומקסימום של המסך ואת המשחק , בנוסף מאתחל את RIGHT ל TRUE

אפרט עכשיו על הפונקציות הייחודיות של במחלקה :

```
public void addEnemy(Block e, double dt)
```
מקבלת בלוק , הופכת אותו לאוייב ומוסיפה לרשימה .

```
public void restart()
```
מורה לכל אויב ברשימה לחזור למצבו המקורי על ידי הפעלת מתודת RESET אצלו.

```
public void removeEnemy(Block e)
```
פונקציה המקבלת בלוק ומחפשת אויב שזהה לו על ידי הפעלת SAME בכל אויב וכשנמצא – מוציאה אותו מהמשחק ומהרשימה.

```
public List<Enemy> getEnemys()
```
מחזירה את רשימה האויבים

```
private Enemy searchstart()
```
מחזירה את האויב הנמצא הכי משמאל

```
private Enemy searchEdge()
```
מחזירה את האויב הנמצא הכי בימין

```
public boolean change(double mx)
```
,מקבלת ערך Y מקסימלי (עד המגנים) מורה לכל האויבים לעשות לעצמם CHANGE וולאחר כל שינוי בודקת אם עברנו את הערך המקסימלי ,במידה וכן מחזירה FALSE אחרת מחזירה TRUE

```
public boolean move(double mx)
```
אחראית על התזוזה ,מקבלת ערך Y מקסימלי (עד המגנים) . אם RIGHT דלוק ניקח את האויב הימני ביותר על ידי SEARCHEDGE ונבדוק דרכו אם התזוזה חוקית (על ידי inrightrenged) במידה והיא חוקית נאמר לכל האויבים ברשימה לזוז ימינה
במידה ואינה חוקית נכבה את RIGHT נבצע CHANGE עם הערך המקסימלי כפרמטר ונחזיר את הערך שchange מחזירה.
אם RIGHT כבוי נבצע את אותו אלגוריתם רק עם תזוזה שמאלה בדיקת הקצה תעשה עם האויב של searchstart מחזירה וכמובן אם התזוזה אינה חוקית נבצע CHANGE ונדליק את RIGHT

```
private Enemy searchEnd(List<Enemy> col)
```
מחזיר אויב במיקום הנמוך ביותר מרשימת אויבים

```
private List<List<Enemy>> setcols()
```
מחזירה רשימה של טורי הבלוקים הקיימים כרגע.

```
private List<Enemy> lastRow(List<List<Enemy>> cols)
```
מקבלת רשימה של טורי בלוקים מתוך כל טור בעזרת searchend מוציאה את הנמוך ביותר וומוסיפה לרשימת בלוקים אותה לבסוף מחזירה. .

```
private Enemy pick(List<Enemy> enemies)
```
מקבלת רשימת בלוקים (שורה אחרונה) ובוחרת רנדומלית בלוק מתוכם ומחזירה אותו.

```
public void timedShoot()
```
יריות בפועל , בעזרתם אלגוריתם זהה לזה של השחקן אני מאפשרת יריה רק כל 0.5 שניות.
בעזרת setcols יוצרת טורי בלוקים אותם אני אשלח לlast row
כאשר הירי חוקי , אני יוצרת רשימת יורים פוטנצייאלים בעזרת lastrow שולחת אותה ל Pick ולאותו האויב אני מורה לירות .

בתוך GAMELEVEL יש לי ממבר של ENEMIES המאותחל בבנאי של GAMELEVEL על ידי מהירות הנשלחת מGAMEFLOW .

```
private void setEnemyies()
```
בתוך הפונקציה הזו אני מסירה את כל האויבים מהמשחק
מבצעת move , במידה והערך המוחזר הוא false אני מורידה חיים ומאתחלת את האויבים למצבם המקורי.
מוסיפה את האויבים בחזרה למשחק (מסירה,מזיזה,מחזירה).

בתוך ה DOONEFRAME -
אני קוראת לsetenemies – מזיזה את כל האויבים וכן קוראת ל timedshoot שתבצע ירי במידה ומתאפשר.

כאשר בלוק נפגע ומודיע על פגיעה מתבצעת בדיקה אם הוא אויב , ורק אם כן הוא יורד מספירת הבלוקים שנשארו כך בעצם המשחק יגמר אך ורק כשכל האויבים יוסרו || נגמרים החיים.

חזרה לgameflow במידה והסתיים משחק נבדוק האם נגמרו החיים :
במידה וכן ← נריץ מסך לוזר ובמידה והתוצאה מספקת נוסיף אותה לטבלת התוצאות ,ונעבור לתפריט הראשי

במידה ולא ← נגביר את המהירות ונתחיל משחק חדש .

רשימת מחלקות :

```
CountdownAnimation
```
אחראית על יצירת אנימציית ספירה לאחור.

```
EndScreen
```
יצירת אנימציית סוף משחק.

```
GameLevel
```
יצירת משחק על פי הוראות .

```
HighScoresAnimation
```
יצירת אנימציית טבלת השיאים.

```
KeyPressStoppableAnimation
```
מגדירה אנימציות כברות עצירה על ידי מפתח מסויים

```
LooserScreen
```
מסך הלוזר

```
PauseScreen
```
יוצרת את מסך העצירה

```
BaseBlockCreator
```
יוצר יוצר בלוקים אשר על פי בקשה יכול להפיק בלוקים מאותו הזן.

```
BlocksDefinitionReader
```
קורא קובץ טקסט של בלוקים מפרש אותם עד כדי יצירת יוצרי בלוקים

```
BlocksFromSymbolsFactory
```
מחזיק מעין מילון אשר מתאים בין סמלים ליוצרי הבלוקים המתאימים להם

```
Filled
```
מנהל את כל עניין מילוי הבלוקים\רקעים בין אם בתמונה בצבע ...

```
LevelSpecificationReader
```
מתרגם קובץ טקסט של תיאור שלב לקובץ של הוראות שניתן להזין למשחק.

```
LevelTranslator
```
מקבל רשימת סטרינגים וממיר אותם לקובץ הוראות שניתן להזין למשחק

```
AnimationRunner
```
מריץ אנימציות

```
Counter
```
מונה

```
LevelsIndicator
```
מורה על שם השלב

```
LivesIndicator
```
מורה על כמות החיים

```
ScoreIndicator
```
מורה על התוצאה

```
Ball
```
יצירת כדור , ציורו , הוזזתו, הוספתו למשחק והוצאתו מן המשחק.

```
Block
```
יצירת בלוק, ציורו, הוספתו והוצאתו מן המשחק.

```
Enemies
```
ניהול כל ממשק האוייבים במשחק

```
Enemy
```
יצירה,ציור,הוזזה,הוספה והוצאה.

```
HighScoresTable
```
יצירה\טעינה של טבלת שיאים ועדכונה

```
Paddle
```
יצירה ציור הוזזה הוספה

```
CollisionInfo
```
מקבץ פרטי התנגשות

```
GameFlow
```
מריץ מהלך של משחק , עד כדי הפסד ,מריץ מסך הפסד, טבלת שיאים וכו

```
GameLevelEnvironment
```
מחזיק רשימה של כל האייטמים ברי ההתנגשות במשחק

```
ScoreInfo
```
מחזיק שם ותוצאה

```
SpriteCollection
```
מחזיק את כל האייטמים המצויירים במשחק

```
Velocity
```
מגדיר, ומחשב מהירות

```
Line
```
קו

```
Point
```
נקודה

```
Rectangle
```
מלבן

```
BallRemover
```
מסיר כדור מהמשחק

```
BlockRemover
```
מסיר בלוק מן המשחק

```
ScoreTrackingListener
```
מעדכן תוצאה

```
ColoredBackground
```
יוצר אייטם בעל צבע אחד

```
CreateLevel
```
יוצר שלב

```
ImageBackground
```
יוצר אייטם עם תמונה כרקע

```
LevelSets
```
סטים של שלבים

```
MenuAnimation<T>
תפריט
```

```
SingleSet
```
סט שלבים אחד

```
Looser
```
מסך הלוזר

```
Screen
```
מסך גנרי

```
Winner
```
מסך מנצח

```
Ass7Game
```
קליטת פרטי משחק הקמת תפריט וביצועו

interfaces

```
Animation
```
אנימציה

```
BlockCreator
```
יוצר בלוקים

```
Collidable
```
בעל יכולת התנגשות

```
Fill
```
בר מילוי

```
HitListener
```
מודיע על פגיעה

```
HitNotifier
```
בעל מודיעים

```
LevelInformation
```
תנאי שלב

```
Sprite
```
בר ציור

```
Menu<T>
```
תפריט

```
Task<T>
```
משימה