תכנות מונחה עצמים תרגיל 3.2

לאחר שבתרגיל 3.1 מימשתם את המשחק Arkanoid, בתרגיל זה תממשו בונוסים ללבנים ("power ups").

הוראות התרגיל

עליכם לממש את ההתנהגויות הבאות:

1. התנהגות רגילה - ההתנהגות אותה ממשתם כבר בחלק הראשון (הלבנה נשברת ונעלמת).
2. כדורים נוספים
3. דיסקית נוספת
4. שינוי מצלמה
5. החזרת פסילה
6. התנהגות כפולה

התנהגות - כדורים נוספים

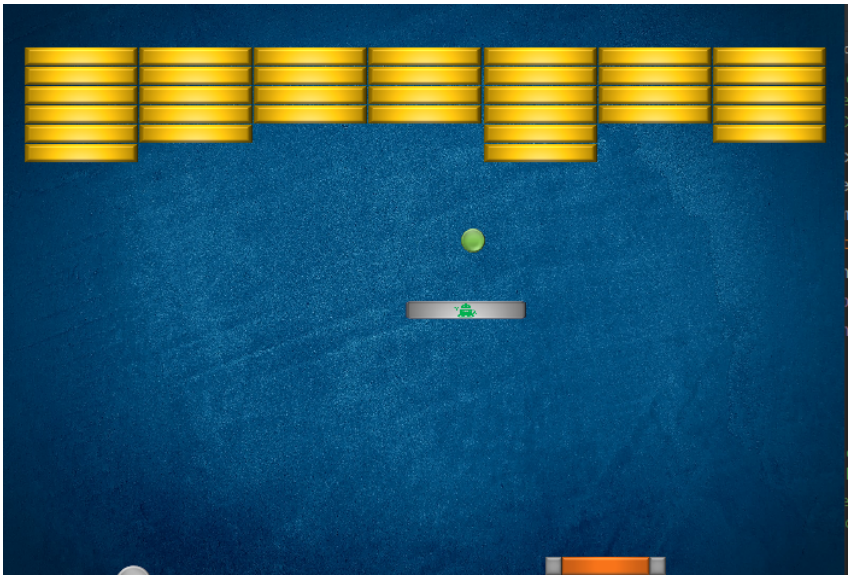
בעת שבירת לבנה בעלת התנהגות כדורים נוספים, יווצרו במרכז המיקום של הלבנה (במקומה) 3 כדורים לבנים (Puck), גדלם שליש מאורך הלבנה והכיוון ההתחלתי של כל אחד מהם יהיה אלכסון רנדומלי. הכדורים יכולים להתנגש ב-pucks האחרים, בדסקית, בלבנים בקירות וכן בכדור המרכזי עצמו (ball).
עבור ה-Puck נשתמש באותו סאונד של הכדור, ובתמונה mockBall.png.

אם Puck יצא מגבול המשחק, הוא אינו משפיע על כמות הפסילות של השחקן.

התנהגות - דיסקית נוספת

בעת שבירת לבנה בעלת התנהגות דיסקית נוספת, תיווצר דיסקית באותה גודל של הדיסקית הרגילה. היא תיווצר במרכז ציר ה-X של המסך, ובגובה של חצי מהמסך (ר׳ התמונה שלמטה).
הדיסקית הנוספת תזוז גם היא לפי תנועות השחקן בדומה לדיסקית המקורית.
שימו לב לנקודות הבאות:

- הדיסקיות לא חייבות להיות מקבילות. כלומר, אם הדיסקית המקורית הייתה צמודה לקיר השמאלי, ואז הופיעה הדיסקית המשנית באמצע, כשנלחץ ימינה ונביא את הדיסקית המקורית לאמצע המסך, הדיסקית המשנית תגיע לקיר הימני. אם נמשיך ללחוץ ימינה רק הדיסקית המקורית תמשיך לזוז כיוון שהדיסקית המשנית כבר הגיע לקצה המשחק.
- בכל רגע נתון יכולה להיות רק דיסקית נוספת אחת. כלומר, סה"כ יכולות להיות רק שתי דיסקיות ברגע נתון במשחק. אם כבר קיימת דיסקית משנית, לא ניצור חדשה.
- לאחר 3 פגיעות בה, הדיסקית תיעלם מהמסך.
- עבור הדיסקית הנוספת, נשתמש בתמונה botGood.png.



התנהגות - שינוי מצלמה

בעת שבירת לבנה המחזיקה בהתנהגות זו, מצלמת המשחק תעקוב אחרי הכדור עד שהכדור פגע ב-4 דברים.
כדי לשנות את המצלמה למעקב אחרי הכדור, נשתמש בשורות קוד הבאות:

```
gameManager.setCamera(
    new Camera(
        ball,                          //object to follow
        Vector2.ZERO,                  //follow the center of the object
        windowController.getWindowDimensions().mult(1.2f),  //widen the frame a bit
        windowController.getWindowDimensions()   //share the window dimensions
    )
);
```

כדי לאתחל חזרה את המצלמה, נשתמש בקוד הבא:

```
gameManager.setCamera(null);
```

שימו לב לנקודות הבאות:

- המצלמה תעקוב רק אחרי הכדור המקורי (Ball ולא Puck), ורק כאשר הוא התנגש ב-4 עצמים, המצלמה תחזור לקדמותה.
- התנהגות המצלמה תופעל רק במידה וההתנהגות לא פעילה כבר (ניתן לבדוק ע"י gameManager.getCamera() == null)

- כדי לעקוב אחרי כמות ההתנגשויות של הכדור, נוכל ל**ממש** פונקציה בשם לדוגמא ball.getCollisionCount המחזירה את כמות ההתנגשויות הנוכחית שביצע הכדור. מומלץ ליצור מחלקה שתטפל במעקב אחרי ההתנגשויות ותבקש לכבות את המצלמה כאשר הכדור התנגש מספיק פעמים.
ניתן למנות את כמות ההתנגשויות של הכדור ע"י מונה הגדל בכל התנגשות שהכדור מזהה.

התנהגות - החזרת פסילה

בעת שבירת לבנה עם התנהגות זו, ממרכז הלבנה ייפול אובייקט לב, אשר על הדיסקית "לאסוף" על מנת להחזיר פסילה.
שימו לב לנקודות הבאות:

- כמות הפסילות המקסימלית הינה 4 (כאשר עדיין הכמות ההתחלתית היא 3). ניתן לשנות את הקוד של Graphic/NumericLifeCounter כדי להתאים את המחלקות להתנהגות החדשה.
- הלב יפול במהירות קבועה של 100 בכיוון ציר ה-Y (כלומר רק למטה).
- הלב יכול להתנגש רק עם הדיסקית המקורית (ולא הדיסקית הנוספת). כדי לאפשר זאת ניתן לדרוס את

הפונקציה shouldCollideWith שקיימת ל-gameObject (שאותו מחלקת הלב תירש). פונקציה זו מאפשרת להגדיר בעת זיהוי התנגשות האם ההתנגשות חוקית.

- כדי לוודא שרק הדיסקית המקורית מתנגשת עם הלב, ניתן להשתמש ב-instanceOf (יש לשים לב שאם הדיסקית המשנית יורשת מהדיסקית המקורית, ה-instanceOf יחזיר אמת, ולכן יש להוסיף בדיקה שגם לא instanceOf של הדיסקית המשנית).
- אם הלב לא מתנגש עם הדיסקית ויוצא מרצפת המסך, יש להסירו מהמשחק.
- גודל הלב הנופל יהיה כגודל הלב שהגדרתם ב-GraphicLifeCounter.

התנהגות כפולה

לבנה בעלת התנהגות זו תגריל התנהגות מבין 5 ההתנהגויות האפשריות (כולל עוד שכבה של התנהגות כפולה, וללא ההתנהגות ה"רגילה").

הערות:

- ללבנה נתונה יהיו לכל היותר 3 powerups - למשל, לבנה תחזיק התנהגות של כדורים נוספים, התנהגות של דיסקית נוספת והתנהגות של החזרת פסילה, ולבסוף יהיה לה את ההתנהגות הבסיסית של הסרת לבנה.
- התנהגות מסוימת יכולה לחזור על עצמה (למשל פעמיים תיבחר התנהגות של כדורים נוספים).
- בסה"כ:
  i. כל אחת מההתנהגויות (רגילה, אחת המיוחדות או הכפולה) תורגל בהסתברות ⅙
  ii. אם מוגרלת ההתנהגות הכפולה, ניתן להגריל רק את אחת המיוחדות (כולל הכפולה, ללא הרגילה), כל אחת בהסתברות ⅕.
  iii. הכפולה יכולה גם להגריל כפולה נוספת, אבל לכל היותר 3 התנהגויות בסה״כ.

טיפים להוספת התנהגויות למשחק

בתרגיל זה יש לכם מעט חופש עיצובי, ולא נגדיר לכם API שאותו יש לממש. יש לכם את כל הכלים שלימדנו כדי לממש את התוספת למשחק.

לעומת זאת, ישנם כמה קריטריונים לעיצוב מתקבל:

1. המנעו מריבוי מחלקות מיותרות. ישנם כלים שלמדתם שימנעו מהצורך ליצור מחלקה לכל שילוב של כוחות אפשרי, ולכן עיצוב כזה אינו תקין.
2. חישבו על API מינימלי ו-access modifiers מתאימים - לא כל מחלקה צריכה להיות פומבית, חישבו על חלוקה מתאימה לחבילות, ועל מגדירי נראות לכל פונקציה ואיבר במחלקות שלכם.
3. עץ ירושה מסועף - את התרגיל ניתן לפתור בעזרת עץ ירושה שאינו גבוה (כלומר אין צורך באבא->בן->נכד->נין וכו'). ניתן להשתמש בכלים שלמדתם בקורס (ירושה, ממשקים, הכלה וכו'). יש להסביר את השימוש בהם ב-README.
4. בכלליות, את עקרונות העיצוב שלכם תצטרכו להסביר ב-README (פירוט בהמשך), ולכן חישבו על קריאות הקוד, אפשרות התחזוק שלו והרחבתו כאשר אתם מעצבים את הפיתרון שלכם.
5. מותר **להרחיב** את הAPI הקיים, בתנאי שתתעדו בREADME את הסיבה להרחבה, וכיצד זה תורם לעיצוב הפיתרון שלכם.

כעת ננסה לתת מעט הכוונה לעיצוב אפשרי:

- את ההתנהגות הבסיסית כבר יצרתם. חשבו על הדרך הנכונה להגדיר את ההתנהגויות הנוספות, מה מבנה המחלקות שלהן, מה ה-API שלהן, ואיך יוצרים אובייקטים שלהן.
- שימו לב שמחלקה של לבנה אמורה לייצג לבנה, ולא התנהגות.
- חשבו כיצד ניתן ליצור כמה התנהגויות במקביל ללבנה אחת. נסו לבנות את התוכנית כך שיהיה קל להרחיב אותה לדרישות עתידיות. כך למשל, אם תתבקשו להרחיב את התוכנה כך שיהיה ניתן ליצור לבנה עם יותר מ-3 התנהגויות, תוכלו לעשות את השינוי במינימום שינויי קוד.
- נדגיש שוב, ההכוונה מדברת על פיתרון אפשרי, ולא הפיתרון המחייב. אם יהיה הסבר מספק לפיתרון שתספקו, הוא יתקבל.

הוראות README

כבכל תרגיל בקורס, יש לעקוב אחרי הוראות הגשת README.

בנוסף להסבר קצר על כל מחלקה בפיתרון שלכם, יש להוסיף את ההסברים הבאים:

1. הסבירו כיצד תכננתם כל התנהגות שמימשתם – רשימות מחלקות, תבניות עיצוב שהמחלקות מממשות, בין אם בעקבות החלטתכם ובין אם בעקבות התכנון של DanoGameLab.
2. אם מדובר בתבנית עיצוב שאתם בחרתם – מדוע בחרתם בתבניות עיצוב אלו? מדוע היה עדיף להשתמש בהן מאשר לתכנן את הקוד אחרת.
3. כיצד הגבלתם את כמות ההתנהגויות הכפולות ל-3 התנהגויות?
4. מה היה העיצוב שבחרתם כדי שלבנה תוכל להחזיק יותר מהתנהגות אחת? כיצד הפיתרון תומך בהרחבה של כמות גדולה יותר של התנהגויות בלבנה אחת?

אופן ההגשה

1. הגישו את התרגיל כקובץ jar בשם ex3_2.jar.
2. בתוך קובץ זה, ימצאו הקבצים הבאים:
   a. הספרייה src, המכילה את קבצי הפיתרון שלכם. בכל קובץ תופיע רק מחלקה אחת ששמה כשם הקובץ.
   b. קובץ בשם README לפי ההוראות לעיל.
3. אין להגיש את קבצי ה-assets.

זכרו את ה-Coding Style של הקורס, והיצמדו להוראות.

בהצלחה!