

תכנות מונחה עצמים – תרגיל 2

תאריך הגשה : 30.11.2025 בשעה 23:55

תרגיל זה מומלץ להגish בזוגות.

0. הקדמה

בתרגיל זה עליים למשתמש משחק דמי "Bricker" שנקרא לו "[Arkanoid](#)".

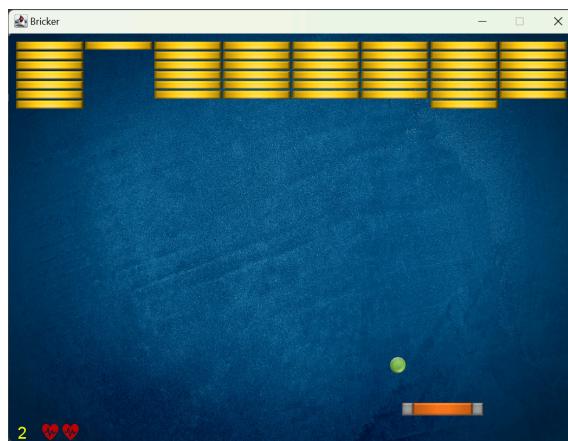
לצורך שימוש זה, אנחנו מספקים לכם מונו-משחק בדמות הספריה 1.1.0 (DanoGameLab). בעזרה ספריה זו ניתן ליצור עצמים של המשחק (`GameObjects`) שאוטם ינהל המנווע באמצעות מנהל המשחק, (`GameManager`). בתרגיל זה אתם תתוכנו את המשחק באמצעות המנווע הזה.

בתרגיל זה נעבד בשני שלבים, בחלק הראשון נבנה משחק בסיסי ובחלק השני נוסף למשחק אפקטים מיוחדים. מומלץ לעבור על כל הוראות התרגיל במלואן לפני תחילת העבודה. החלק הראשון של התרגיל ניתן בקורס מודרנת ומולוה בסרטונים. חלק זה יעזור לכם למדוד איך לעבוד עם הספריה של DanoGameLab. בנוסף לסרטונים באתר, תוכלו למצוא במודול גם [מדריך](#) מיקוצר לעובדה עם הספריה. בחלק השני של התרגיל תצטרכו אתם לחשב על העיצוב ולהחליט בעצמכם אילו מחלקות ליצור ובאיו תכניות עיצוב להשתמש בהתאם לעקרונות שלמדנו.

המשחק צפוי להיראות בערך כה:

חלק ב- המשחק המורחב:

חלק א- המשחק הבסיסי:



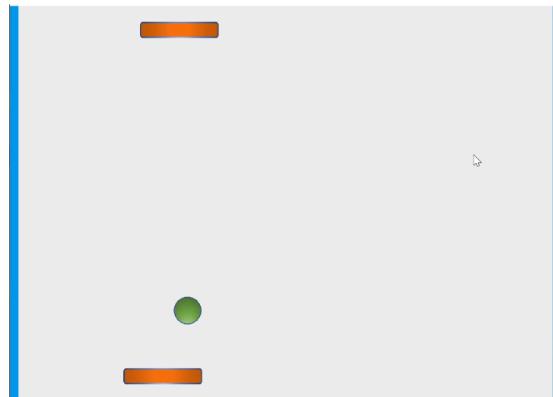
במהלך המשחק השחקן מזיז את הדיסקית שזזה שמאלה וימינה, הcador צריך לשבור את כל הלבנים, ואסור שהcador יפול אל מעבר המסך.

במשחק הבסיסי: כשלבנה נשברת היא נעלמת זהה. כשלון נשברו - ניצחנו.
במשחק המורחב: כל מיני דברים עשויים לקרות כל פעם שלבנה נשברת: יתכן שיופיעו כדורים מסוימים, דיסקית נוספת, ייפלו חיים נוספים, או שהcador ישנה מהירות.

שימוש! בתרגיל זה צפויים לכם שני אטגרים חדשים, גם למדוד איך לעבוד עם ספריה חדשה, וגם בפעם הראשונה בקורס אתם צריכים לחשב על עיצוב הקוד בחלק מהתרגיל בעצמכם!

1. חלק ראשון

בחלק זה של התutorial נבנה את המשחק הבסיסי. המדריך של חלק זה יתבסס על סדרת סרטונים שעוסקים בהקנת משחק דומה ל-Pong שנקרא לו Bricker, והוא נראה כך:



גם ב-game Ping המשחק בדיסקיית התחתונה אסור לתת לכדור ליפול, גם כאן הcador מקפץ וכן הלאה, רק שבמוקם לבנים שנשברות בחולק העליון, יש שם דיסקית יירבה שימושה המחשב. אם דיסקית היריב נזנחה לכדור לחוף על פניה אל מעל למסך, השחקן ניצח. אנחנו נעקוב אחרי סדרת הסרטונים ובגדל נועתייק ממנו באופן גס, פרט להבדלים שיציינו לצד כל סרטון.

היכנסו לאתר הקורס במודול, תרגיל 2, [מדריך לחלק א של התutorial](#) ועקבו אחר ההוראות ליצירת המשחק- מהמדריך להתקנת הספריה ועד לסיום תת הפרק. **שים לב למספר הסעיף בראש כל לשונית לצורך מענה על שאלות בפורים.**

המבנה של ה-API של התutorial שלכם בסוף החלק הראשון צריך להיות לפי ההגדרות הבאות (שים לב למצויינים ב-API רק שדות ושיטות שהם public):

package bricker.gameobjects;

1) Class Ball

- **constructor**
- **void onCollisionEnter(GameObject other, Collision collision)**

2) Class Paddle

- **constructor**
- **void update(float deltaTime)**

3) Class Brick

- **constructor**
- **void onCollisionEnter(GameObject other, Collision collision)**
- **Methods for accessing a brick's position in the grid (row and column)**

4) A class or classes that implement a graphic and a numeric life counters.

package bricker.brick_strategies:

5) Interface CollisionStrategy

- void **onCollision**(GameObject thisObj, GameObject otherObj)

6) Class BasicCollisionStrategy

- **constructor**
- void **onCollision**(GameObject thisObj, GameObject otherObj)

package bricker.main:

7) Class BrickerGameManager

- public static void **main**(String[] args)
- **constructor**
- void **initializeGame**(ImageReader imageReader, SoundReader soundReader, UserInputListener inputListener, WindowController windowController)
- void **update**(float deltaTime)
- **Possibly more methods according to your design.**

הערות חשובות:

1. הסרטונים במדרייק הם קטעים חתוכים מתוך סרטוני יוטיוב וביניהם קטעי טקסט עם הוראות. לבטיחותכם והנאתכם, צפוי בהם דרך המודול, לא דרך יוטיוב.

2. הגדים והמיקומים המופיעים בסרטונים גם בגדיר המלצה. אתם יכולים להתאים את הקבועים כרצונכם (כגון מהירות הcador והדיסקית) כדי לשפר את חווית המשחק או אם תראו בכך צורך בשביל החלק השני. לא תידקנו על קבועים מדויקים. אם זאת, אין להגדיר קבועים קיצוניים כך שחוויות המשחק תפגע או שלא יהיה ניתן לעשות פעולות מסוימות, זה יכול לפגוע בನיקוד בGRADE ולא ניתן לבדוק את המשחק שלהם.

2. חלק שני:

2.1. התנהוגיות של הלבנים

לאחר שבחلك הראשון מימושם את המשחק Bricker לפי המדריך במודול, בחלק זה תמשו התנהוגיות מיוחדות לחלק מהלבנים שיתרחשו כאשר הcador יתגש בלבנים. כל הלבנים במשחק יעלמו כאשר הcador פוגע בהם, אך חצי מהלבנים יגרמו לדברים נוספים לקרות כאשר הונעלמות.

הערה חשובה! קראו את התרגיל עד הסוף לפני שאתם מתחילה בימוש!

בחלק הזה של התרגיל ניתן לכם חופש עיצוב, אך אם עברתם על כל התרגיל מראש, ועיצבתם נכון את המחלקות שלכם בהתאם לעקרון Open-Close, השינויים בחלק הזה צפויים להיות רק בהוספת מחלקות ושינויים למנהל המ משחק, ככלומר לא אמרו להיות צורך בשינויים ב-API של המחלקות שיצרתם בחלק הראשון מלבד המחלקה `BrickerGameManager`.

התנהוגיות של הלבנים יהיה לפי ההתפלגות הבאה (הסביר מפורט של התנהוגיות המיוחדות בחלקים הבאים):

- בסתירות ½ : התנהוגות רגילה - התנהוגות אותה מימושם כבר בחלק הראשון (הלבנה נשברת ונעלמת).
- בסתירות 1/10: כדורים נוספים.
- בסתירות 1/10: דיסקית נוספת.
- בסתירות 1/10: לבנה מתפוצצת.
- בסתירות 1/10: החזרת פסילה.
- בסתירות 1/10: התנהוגות כפולה.

סה"כ בתחילת, חצי מהלבנים יהיו רגילים וכחציתם התנהוגות מיוחדת, כאשר לכל אחת מההתנהוגיות המיוחדות תהיה הסתברות שווה להופיע במהלך המשחק.

2.2. התנהוגיות מיוחדות

2.2.1. כדורים נוספים

בעת שבירת לבנה בעלת התנהוגות זו, יוצרו במרכז המיקום של הלבנה (במקומה) 2 כדורים לבנים להם נקרא "Puck". התוכנות של כל Puck:

- 2.2.1.1. התמונה שלו היא `mockBall.png` הנמצאת בקובץ `assets`.
- 2.2.1.2. משתמש באותו אונד של הcador הרגיל.
- 2.2.1.3. גודלו ¼ מגודל הcador הרגיל גם באורך וגם ברוחב.

הכוון ההתחלתי של כל כדור Puck יהיה כיוון רנדומלי על החצי העליון של מעגל היחידה.
ניתן לעשות זאת באמצעות:

```
Random random = new Random();
double angle = random.nextDouble() * Math.PI;
float velocityX = (float) Math.cos(angle) * BALL_SPEED;
float velocityY = (float) Math.sin(angle) * BALL_SPEED;
```

- .2.2.1.5 הוא יכול להתנגש ב-pucks האחרים, בדיסקית (Paddle), לבנים בקירות וכן בכדור המרכז עצמו (ball).
- .2.2.1.6 אם הוא יצא מגבול המשחק, הוא אינו משפיע על כמות הפסילות של המשחק ויש לדאוג להסיר אותו מראשית האובייקטים של המשחק.
- .2.2.1.7 שימושו ללב Ci יכול להיות מצב בו שני כדורים פוגעים לבנה אחת באותו הזמן. על מנת לבצע ספירה וכוננה של gameObjects().removeGameObject()

2.2.2. דיסקית נוספת (Extra Paddle)

בעת שבירת לבנה בעלת התנהגות זו, תיווצר דיסקית נוספת על המסר. התכונות של הדיסקית הנוספת:

- .2.2.2.1 זהה לדיסקית הרגילה (אותו גודל ואותה תמונה של הדיסקית הרגילה).
- .2.2.2.2 היא תיווצר במרכז ציר ה-X של המסר, וגובהה של חצי מהמסר.
- .2.2.2.3 היא תזוז לפי תנועות המשחקן בדומה לדיסקית המקורי.
- .2.2.2.4 הדיסקיות לא חייבות להיות באותו מקום על ציר ה-X. כמובן, אם הדיסקית המקורית הייתה צמודה לקיר השמאלי, אז הופעה הדיסקית המשנית באמצע, כשלוחץ ימינה ונביא את הדיסקית המקורית לאמצע המסר, הדיסקית המשנית תגיע לקיר הימני. אם נמשיך לחוץ ימינה רק הדיסקית המקורית תמשיך לזרז כיוון שהדיסקית המשנית כבר הגיעו לקצה המסר.
- .2.2.2.5 ככל רגע נתון יכולה להיות רק דיסקית נוספת אחת. כמובן, סה"כ יכולות להיות רק שתי דיסקיות ברגע נתון במהלך משחק. אם כבר קיימת דיסקית נוספת, לא ניתן חדשה.
- .2.2.2.6 לאחר 4 פגימות בה, הדיסקית הנוספת תעלם מהמסר.

2.2.3. לבנה מתפוצצת

בעת שבירת לבנה המחזיקה בתנהגות זו, הלבנה "מתפוצצת" ושבורת לבנים שמסביבה:

- .2.2.3.1 יונגן הסאונד מהקובץ assets/explosion.wav.
- .2.2.3.2 שבירה של הלבנה תוביל לשבירה של הלבנים הסמוכים אליה (לא כולל אלכסונים). כמובן שבירה של לבנה בשורה ה-i- ב العمودה ה-j תוביל גם לשבירה של הלבנים באותו מקום ה-i (אם יש לבנה באותו מקום זה):

$$[i-1, j], [i, j], [i+1, j], [i, j-1], [i, j+1]$$
- .2.2.3.3 במדיה ולבנים עם התנהגות מיוחדת נשברו עקב יכולת זו, התנהגות המיוחדת תפעל zusätzlich. לדוגמה, אם המשחקן שבר לבנה מתפוצצת שכנה לבנה מתפוצצת רצף הפעולות הבא יקרה:
- הלבנה שהכדור פגע בה מתפוצץ

- כל הלבנים השכנות של הלבנה המתפוצצת ישברו, ובפרט הלבנה המתפוצצת השכנה המתפוצץ
- הלבנים השכנות של הלבנה השכנה יתפוצצו וכן הלאה.

הכוונה לעיצוב:

- 2.2.3.4. שימוש לב, לבנה כבר יש קטע קוד שאומר "תעשי X כשפוגעים בר" שהופעל ע"י מנוע המשחק אוטומטית בכל מקרה של התנגשות.

2.2.4. החזרת פסילה

בעת שבירת לבנה המחזיקה בהתנוגות זו, ממרכז הלבנה ייפול אובייקט לב, אשר על הדיסקית "לאסוף" על מנת להחזיר פסילה. כלים של החזרת פסילה:

- 2.2.4.1. גודל הלב הנופל יהיה כפול הלב שהגדרתם בהציג מספר החיים וישתמש באוותה התמונה.
- 2.2.4.2. הלב הנופל יופיע באותו מקום של מרכז הלבנה שנעלמה, ויפול במהירות קבועה של 100 בכיוון ציר ה-Y (כלומר רק למטה).
- 2.2.4.3. כמות הפסילות המקסימלית במשחק תהיה 4 (כאשר עדין הכמות ההתחלתית היא 3). יתכן וצטרכו בשלב זה לשנות את הקוד של המחלקות (או המחלקה) שemmמשות את הצגת מספר החיים, כדי להתאים את המחלקות להתנוגות החדשה.
- 2.2.4.4. הלב הנופל יכול להתנגש רק עם הדיסקית המקורית (ולא הדיסקית הנוסף). אם הלב לא מתנגש עם הדיסקית יויצא המסך, יש להסירו מרישימת האובייקטים של המשחק.

הכוונה לעיצוב:

- 2.2.4.5. כדי לדאוג שהלב מתנגש רק עם הדיסקית המקורית ניתן לדרוס את הפונקציה `shouldCollideWith` שקיימת ל-`gameObject` (שאותו מחלוקת הלב תירש). פונקציה זו מאפשרת להגדיר בעת דיזי התנגשות האם התנגשות חוקית.
- 2.2.4.6. חשבו איך יהיה נכון לבדוק שאכן התנגשותם בדיסקית המקורית בלי שימוש ב- `instanceOf` ובלי עיקפת פולימורפיים.

2.2.5. התנוגות כפולות

עבור לבנה בעלת התנוגות זו יוגלו שתי התנוגויות מיוחדות מבין 5 התנוגויות האפשריות (כולל עוד התנוגות כפולות בהתבגרות שווה לכל אחת, והיא תפעיל את שתיהן כאשר הcador מתנגש בה).

כלים של התנוגות כפולות:

- 2.2.5.1. לבנה נתונה יהיה **לכל היוטר 3 התנוגויות מיוחדות**. למשל, לבנה תחזיק התנוגות של כדורים נוספים, התנוגות של דיסקית נוספת והtnogot של החזרת פסילה, (וכמובן תהיה לה את התנוגות הבסיסית של הסרת לבנה כמו כל לבנה).
- 2.2.5.2. **שימוש לב!** מקרה מסוים שבו יש 3 התנוגויות מיוחדות נוצר מכך שהगכנו עבור הלבנה התנוגות כפולות, ואז **לא אחת** מהtnogot המיוחדות היגרנו שוב התנוגות כפולה. לא ניתן להגリアル שוב התנוגות כפולה מעבר לכך.
- הtnogot מסוימת יכולה לחזור על עצמה בהtnogot כפולה, למשל יתכן שתבחר התנוגות של כדורים נוספים פעמיים אותה לבנה (במקרה זה יויצרו 4 כדורים).

טיפ לעיצוב: התנהוגות כפולה היא בעצם התנהוגות מיוחדת חדשה ש"עוטפת" שתי התנהוגות מיוחדות אחרות. השתמשו בתבנית עיצוב מתאימה שלמדו המתאימה למצב זהה. שימוש לב שימוש של התנהוגות אחת לא אמור להיות מושפע מהוספת שימוש של התנהוגות אחרת, ובפרט, התנהוגות המיוחדות האחרות לא אמורים להיות מושפעות מכך שיש התנהוגות שהיא כפולה אלא זו התנהוגות נפרדת.

2.3. קriterוניים והכוונה לעיצוב נכון.

בחילק השני של התרגום זה יש לכם חופש עיצובי מסוים, ולא הגדרנו לכם API שאותו יש למשם. לעומת זאת, ישנו כמה קriterוניים לעיצוב מתקובל:

- .2.3.1. הימנו מרבי מחלקות מיותרות. לדוגמה, יצירת מחלוקת לכל שימוש של התנהוגות אפשרי, הוא לא עיטוב תקין.
- .2.3.2. חישבו על API מינימלי ועל מגדייר נראות לכל פונקציה ואיבר במחלקות שלכם.
- .2.3.3. את התרגום ניתן לפטור בעזרת עץ ירושה שאין גבוה (כלומר אין צורך באבא->בן->נכד->נין וכו'). ניתן להשתמש בכלים שלמדתם בקורס (ירושה, ממוקמים, הכללה וכו'). יש להסביר את השימוש בהם ב-README.
- .2.3.4. חישבו על קרייאות הקוד, אפשרויות התחזוק שלו ורחבתו כאשר אתם מעצבים את הפיתרון שלכם.
- .2.3.5. מותר להרחיב את ה-API של חלק 1 אם אתם יודעים להסביר למה זה נדרש בעיצוב שלכם. אם תבחרו לשנות את ה-API הנוכחי, תתבקשו לתעד ב-README את הסיבה לכך, וכייזד זה תורם לעיצוב הפיתרון שלכם.

cut נסעה לתת מעט הכוונה לעיצוב אפשרי:

- .2.3.6. את התנהוגות הבסיסית כבר צרתם. חישבו על הדרך הכוונה להגיד את התנהוגות הנוספות, מה מבנה המחלוקת שלהם, מה ה-API שלהם, ואייר יוצרים אובייקטים שלהם.
- .2.3.7. חישבו מבחינה עיצובית איפה יהיה המקום בקוד להציג את התנהוגות של כל לבנה.
- .2.3.8. חישבו כיצד ניתן ליצור כמה התנהוגות במקביל לבניה אחת. נסו לבנות את התוכנית כך שהיא קל להרחיב אותה לדרישות עתידיות.
- .2.3.9. חישבו כיצד לעצב את הקוד כדי לכלול את ההגבלת של 3 התנהוגות מיוחדות לבנה. עצבו את הקוד כך שגם תתבקשו להרחיב את התוכנה כך שהיא ניתן ליצור לבנה עם יותר מ-3 התנהוגות, תוכלו לעשות את השינוי במינימום שינויי קוד.
- .2.3.10. זכרו את ה-[Coding Style](#) של הקורס, והיצמדו להוראות.

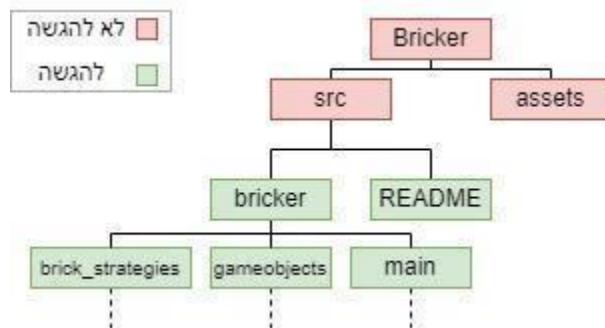
3. הוראות הגשה

- 3.1. יש להגיש מחשבון אחד בלבד.
- 3.2. הגישו את התרגיל כקובץ `jar/tar/zip` בשם **ex2** (והסויימת בהתאם).
- 3.3. בתוך קובץ זה, ימצאו הקבצים הבאים:

3.3.1. התיקייה `bricker` המכילה את קבצי הפתרון שלכם לפי ה-API המפורט בסוף חלק 1, בתוספת המחלקות שהוספتم עבור חלק 2. הקפידו על חלוקה לחבילות. בכל קובץ תופיע רק מחלוקת אחת ששמה כשם הקובץ.

3.3.2. קובץ בשם `README` (ללא סויימת) לפי ההוראות בסעיף 3.4.

3.3.3. ככלומר, זה עץ ההגשה שלuproject כאשר יש להגיש את התקינות והקבצים **ירוקים**:



שימו לב - כל חבילה שהוספتم לקוד צריכה להיות תת חבילה של החבילה `bricker`.

- 3.4. אין להגיש את קבצי `assets` (דאגו לא לשנות את השמות של הקבצים בהם אתם משתמשים בקוד שיהיו זהים לקבצים שהורדתם).

3.5. קובץ README:

3.5.1. בשורה הראשונה בקובץ יופיע שם המשתמש *CSE* שלכם.
אם אתם מגישים בזוג, יש להפריד בין שמות המשתמש עם פסיק (שנייהם באותה שורה). לדוגמה: user1,user2

3.5.2. בשורה השנייה מספר תעודת הזהות.
אם אתם מגישים בזוג, יש להפריד בין מספרי הזהות עם פסיק (שנייהם באותה שורה). לדוגמה:
12345678,87654321

3.5.3. השורה השלישית ריקה.

3.5.4. החל מהשורה הרביעית ענו בקובץ על השאלות הבאות לפי הסדר:

1. כתבו איזה עיצוב בחרתם בחולק 1.7 כדי לאפשר העלמת לבנים במשחק, וסבירו מה הם **ה יתרונות והחסרונות** של הבחירה העיצובית שלכם.
2. הסבירו איך מימושם את הצגת מספר החיים של השחקן (גרפי ומומרי) מחולק 1.8 של התרגיל. הסבירו בקצרה מה תפקידה של כל מחלוקת שהוספتم בקוד.

3. הסבירו בקצרה כיצד מימשتم כל אחת מהתנהוגיות המיווחדות (מלבד הכפולה). הסבירו על תפקידן של מחלקות שהווסףתם כדי למש את התנהוגות, אם הוסיףם.
4. הסבירו כיצד מימשتم את התנהוגות הכפולה בחלק 2.2.5. התיחסו בהסביר לעיצוב של הקוד, וכן כיצד הגבלתם בקוד את כמות התנהוגות ל-3.
5. אם עשיתם בחלק 2 של התרגיל שינויים ב-API שמיימשتم בחלק 1 שלו, הסבירו בקצרה למה השינוי זהה היה הכרחי לעיצוב שלכם.

4. בדיקת הגשה

- 4.1. לאחר שהגשתם את התרגיל בתיבת הגשה הקוד שלכם עברו טסט presubmit וווצר הקובץ pdf.oisnmission..submision. וודאו שהקובץ מוצר בהצלחה.
- 4.2. הקובץ מכיל פלט של הטסט המודד שהקוד שלכם מתקמל, ומפרט על שגיאות בסיסיות. השתמשו בפלט שבקובץ על מנת לתקן שגיאות בתרגיל שימנוו מאייתנו להריץ את הטסטים הסופיים. (זהו טסט קדם הגשה ולא הטסט הסופי של התרגיל).
- 4.3. ניתן להגיש את התרגיל שוב ושוב ללא הגבלה עד למועד הגשה. (הגשה האחרונה היא ההגשה הסופית).
- 4.4. שימוש לב: על פי נהלי הקורס חובה לעבור את הטסט `presubmit` ללא שגיאות. קובץ הגשה שלא עבר בהצלחה את הטסט יקבל ציון 0 ולא "בדק".
- 4.5. ניתן לחולופין להריץ ישירות את ה `presubmit` על ידי הריצת הפקודה הבאה (במחשב בית הספר):
`<oop1/ex2_presubmit <path to your file~`
- 4.6. שימוש לב שפקודה זו לא מגישה את התרגיל בפועל אלא רק מרים את ה `presubmit`.
- 4.7. חובה לעבור תמיד גם על הפלט של `pdf.oisnmission` לאחר הגשה בתיבת הגשה לוודא שהכל תקין!
- 4.8.

בהצלחה!

גופח שאלות נפוצות והמלצות

לפני תחילת העבודה על התרגיל השקיעו זמן בהבנה של הספריה ובפרט איך לעבוד עם מבנה השכבות. עבודה לא יעה עם שכבות יכולה להוביל למשחק איטי שלא רץ טוב עקב חישוב התנגשיות וקריאות מיותרות לפונקציות.

החל מעמוד 10 במורה נבוכים של DanoGameLab יש הסבר מפורט על מבנה השכבות וההתנשויות, מומלץ אפילו להיכנס לקוד של הספריה של דן ולנסות ולהבין בעצמכם איך זה בניו.

- האם ניתן להרחיב את ה- API בחלק 1 מעבר למה שמפורט בתרגיל?

כן, אם יש סיבה עיצובית מוצדקת לכך, אותה יש להסביר ב- README

- האם צריך לבדוק תקינות של הקלט ומה צריך לעשות אם הוא לא תקין?

אם התקבלו שני ארגומנטים ניתן להניח שהם תקינים (כלומר מספרים שלמים וחוביים). בשאר המקרים יש לאותל את המשחק עם ערכים דיפולטיבים.

- האם יש פסילה תוך כדי שההתנשגות מיוחדת מופעלת האם והתנשגות צריכה להתאפס?

לא, כאשר עדין יש חיים ההתנשגות צריכה להמשיך להמשיך, ככלומר לא נדרש מכם לבצע פעולות נוספות כדי להפסיק את ההתנשגות.

- האם הספריה של ההתנשויות כוללת את ההתנגשות בלבנה?

ספריה של התנשויות היא לאחר ההתנגשות בלבנה. שימוש לב שזה תלוי בסדר הקראיה של מנוע המשחק למתחודת .onCollisionEnter

- רשום כי לאחר 4 פגימות בה-paddle הנוסף יעלם מן המשחק, האם פגיעה נחשבת רק פגעה על ידי הcdcור הראשי או גם פגעה על ידי puck?

גם Puck יחשב לספרית הפגימות ב- paddle.

- איך עובד סדר הקראיה של מנוע המשחק למתחודות ?onCollisionEnter

מנוע המשחק מבצע כל פרימס בדיקה עבור התנשויות וקראיה למתחודות onCollisionEnter של האובייקטים שהתנגשו לפי סדר השכבות בהם נמצאים.

כלומר השאלה אם נקרהת המתחודה של אובייקט כזה או אחר קודם תלוי בחלוקת של האובייקטים לשכבות. הדרך פשוטה להתמודד עם זה היא על ידי הרצת ניסיונות ושינוי הקוד בהתאם.

- במקרה שהcdcור הראשי נפסל ולא נשארו חיים, האם רצוי שנמשיך את המשחק עד שה-pucks גם הם יפסלו או האם פסילה של הcdcור הראשי מספיקה כדי לסיים את המשחק גם אם נשארו pucks.

פסילה של הcdcור הראשי מסיימת את המשחק.

- האם הcadour אמור להצליח להתנגן בלבבות הנופלים? ולשנות את הכוון שלו בהתאם?

לא

- אם אנחנו מקבלים התחנחות כפולה של שתי החזרות פשוט, אז לפחות ההנחה נראית רק לב אחד כי הם אחד על השני, האם זה נכון?

כן.

- מה צריך לטעוד? איך מתעדים שדות פרטיים ושדות פומביים? מה נחשב חלק מה- API ?

נא לקרוא את מסמך ה- [Coding-style](#) במודול.

- האם צריך למחוק אובייקטים מרשימה `?gameObjects`

כן. אובייקטים ששינו את תפקידם במשחק צריכים להימחק כדי להימנע מעומס על המENU