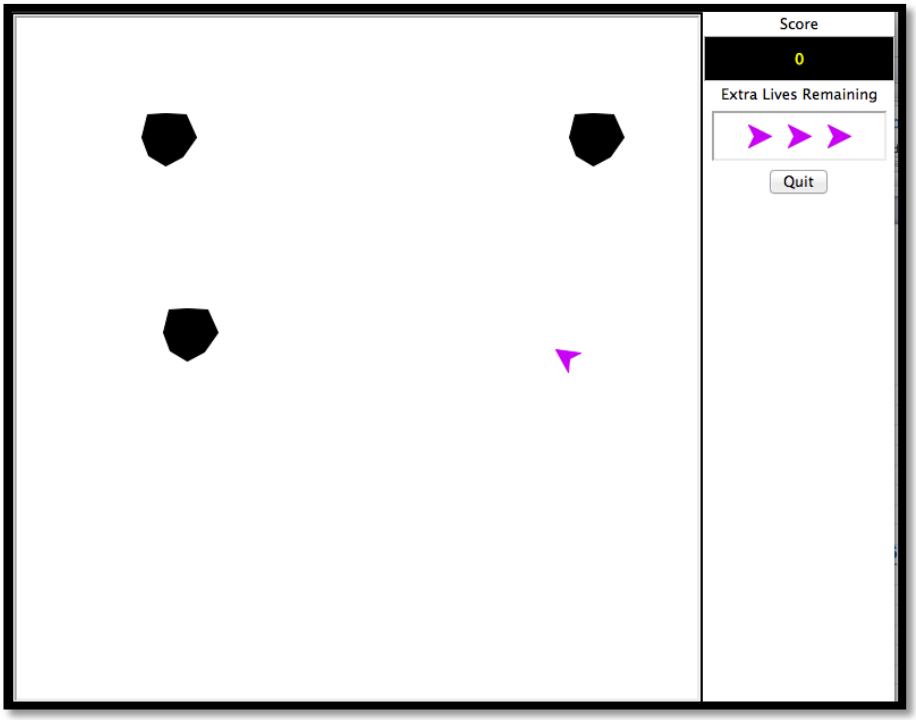
בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

מבוא למדעי המחשב 67101 – סמסטר א' 2020
תרגיל 10 – Asteroids
להגשה בתאריך **31/12/2020** בשעה 22:00

בתרגיל הנ"ל תתבקשו לממש את המשחק **Asteroids** (להסבר נוסף לחצו על הלינק). התרגיל ישתמש במודול שהכרתם בתחילת הקורס בתרגיל הראשון, **turtle**.

בסוף התרגיל אתם תייצרו משחק שייראה כך:



הנחיות כלליות:

- את התרגיל **חובה להגיש בזוגות**. התרגיל ארוך, התחילו לעבוד עליו מוקדם!
- ביצוע המשימות לפי סדר יבטיח פעולה נכונה של המשחק, אך ישנן דרכים שונות בהם ניתן לבצע את התרגיל - המשימות המפורטות בהמשך הם בגדר המלצה בלבד.
- עליכם להשלים את מימוש חלק מהפונקציות בקובץ asteroids_main.py ואף תוכלו להוסיף פונקציות משלכם. בנוסף לכך, ניתן, ואף מומלץ, להוסיף קבצים נוספים משלכם.
- לצורך מימוש התרגיל מימשנו עבורכם את המחלקה Screen הנמצאת בקובץ screen.py, מחלקה זו אחראית על ציור האובייקטים למסך ואתם מחויבים להשתמש בה במהלך כל התוכנית - אין לשנות מחלקה זו (**בפרט אתם לא תגישו אותה**).

הערה: הקובץ screen.py מכיל מחלקה נוספת בשם ShapesMaster, אין לכם צורך לעשות בה שימוש ישיר במהלך כתיבת התוכנית שלכם.

כדי לקרוא יותר על המחלקה Screen ועל הפונקציות שהיא חושפת אתם מוזמנים לפתוח את הקובץ index.html הנמצא בקובץ api.zip.

פתרון בית הספר

ממומשת עבורכם גרסה של המשחק. מומלץ לאחר קריאת התרגיל וטרם תחילת המימוש להריץ את המשחק. המשחק אינטואיטיבי ומספר משחקים בו מבהירים את הכוונה הכללית על ניהול המשחק כמו גם שאלות על פרטים ספציפיים. אם יש לכם ספק בנוגע להתנהגות המצופה בתרחיש מסוים – סביר מאוד להניח שתוכלו למצוא אם התשובה לכך דרך פתרון בית הספר, ולכן אתם מתבקשים לנסות להבין דרכו את הדרישות לפני פניה בשאלה בפורום.
הפיתרון הוא הקובץ המקומפל asteroids_main הנמצא בתיקייה :

~intro2cs2/bin/ex10/

את פתרון בית הספר ניתן להריץ ממחשבי בית הספר בעזרת הפקודה :

~intro2cs2/bin/ex10/asteroids <num_asteroids>

כאשר הפרמטר num_asteroids הינו אופציונאלי.

רקע מקדים

במשחקי מחשב, ובתכנות בכלל, מתרחשות הרבה פעולות בו זמנית, למשל הזזה של העכבר תוך כדי לחיצה על מקשי המקלדת. ישנן שתי גישות למימוש התנהגות שכזו, אקטיבית ופסיבית.

<u>בגישה האקטיבית:</u> ברגע שמתבצעת פעולה ע"י המשתמש (למשל הזזת העכבר) התוכנה תגיב ללא עיכוב.

<u>בגישה הפסיבית:</u> ברגע שמתבצעת פעולה ע"י המשתמש תידלק "נורה" אשר תיבדק באופן מחזורי ע"י התוכנה – באם הנורה דולקת התוכנה תבצע את אותן פעולות שהייתה מבצעת אילו היו מתרחשות בגישה האקטיבית, ותכבה את הנורה.

ההבדל העיקרי בין הגישות הוא ה-"מיידיות" של הפעולה, בגישה הפסיבית אנחנו יכולים להגדיר שנבדוק האם הנורה דולקת כל כמה שניות לעומת הגישה האקטיבית שבה נטפל בכל פעולה מיידית בשנייה בה היא נדרשת.

בתרגיל זה ננקוט בגישה הפסיבית. מכיוון שאנחנו מייצרים משחק ניתן להתייחס לכל הפעולות כאילו הן קורות ברצף קבוע כלשהו (הפעולות שיש לבצע מתוארות למטה). את רצף הפעולות הזה תצטרכו לממש בפונקציה game_loop לפי סדר מסוים, לשיקולכם.

הקובץ asteroids_main.py מכיל מחלקה בשם GameRunner המכילה את הפונקציה game_loop וכן פונקציות נוספות. מחלקה זו היא המחלקה הראשית של המשחק, היא אמורה לייצר את האובייקטים השונים של המשחק, להכיל מימוש של האינטראקציות השונות ביניהן ולדאוג לרצף הפעולות התקין של המשחק. עליכם להשלים את תוכן חלק מהפונקציות של מחלקה זו, ואף תוכלו להוסיף פונקציות נוספות למחלקה, לשיקולכם.

הפונקציה game_loop נתונה לכם כחלק מהמחלקה GameRunner. הפונקציה אחראית על ביצוע הפעולות השונות האחראיות על מהלך המשחק, ונקראת באופן מחזורי שוב ושוב לאורך כל ריצת המשחק. את הפונקציה הזו אתם תצטרכו להשלים לפי המשימות בשלבים א'-ה' (בסדר הזה) הנתונים למטה. מכיוון שמימוש כלל המשימות יחרוג מהאורך המותר של פונקציה בקורס, אנחנו מעודדים אתכם להפעיל שיקול דעת ולכתוב פונקציות נוספות שייעזרו לכם בפתרון כל משימה – שימו לב! יכול להיות שאותה הפונקציה תוכל להועיל לכם ביותר ממשימה אחת. המטרה של התרגיל הינה כתיבת אובייקטים מורכבים וחשיבה על האינטראקציה בין הישויות השונות בתוכנית.

מבנה התרגיל

1. בשלב הראשון תתבקשו לממש את מחלקות האובייקטים המרכזיים במשחק שהם החללית, האסטרואידים והטורפדואים.
2. בשלב השני תתחילו לממש את פונקציית המשחק הראשית (game_loop), שתמשיך להתעדכן גם במהלך השלבים הבאים. אתם תתבקשו לממש את החלקים האחראים על הופעה ותזוזה של החללית. בסיום שלב זה המשחק שלכם יכיל חללית בלבד, אותה תוכלו לסובב ולהזיז (אך עוד לא לירות טורפדואים).
3. בשלב השלישי תתבקשו לממש את החלקים האחראים על הוספה של אסטרואידים וכן אינטראקציות מתקדמות בין החללית לאסטרואידים (כגון התנגשות). בסיום שלב זה המשחק שלכם יכיל אסטרואידים זזים וחללית שיכולה להתנגש בהם.
4. בשלב הרביעי תתבקשו לממש את החלקים האחראים על הוספה של טורפדואים, וכן אינטראקציות מתקדמות בין טורפדו עם החללית והאסטרואידים (כגון פגיעה של טורפדו באסטרואיד). בשלב זה המשחק אמור לעבוד כמצופה.
5. בשלב החמישי (והאחרון) תתבקשו לממש אלמנטים מתקדמים במשחק.

המלצת צוות הקורס היא לבצע את המשימות בסדר שבו הן מופיעות ולבדוק בכל שלב שהמשחק עומד בדרישות המתאימות.

שלב א'

<u>משימה 1 – מימוש מחלקת החללית:</u>

לחללית יש את התכונות הבאות (ייתכן ותכונות נוספות יתווספו בהמשך):

- מיקום ומהירות על ציר ה-x
- מיקום ומהירות על ציר ה-y
- כיוון חרטום (במעלות).

עליכם ליצור קובץ בשם ship.py ובו לממש את המחלקה Ship שתכיל את כל התכונות הללו. התכונות ישמשו אתכם בהמשך למימוש החלקים האחראים על הזזת החללית והאינטראקציות עם האובייקטים האחרים במהלך המשחק.

<u>משימה 2 – מימוש מחלקת האסטרואיד:</u>

לאסטרואיד יש את התכונות הבאות (ייתכן ותכונות נוספות יתווספו בהמשך):

- מיקום ומהירות על ציר ה-x
- מיקום ומהירות על ציר ה-y
- גודל - מספר שלם (integer) בין 1 ל-3

עליכם ליצור קובץ בשם asteroid.py ובו לממש את המחלקה Asteroid שתכיל את כל המידע הרלוונטי על האסטרואיד. התכונות ישמשו אתכם בהמשך למימוש החלקים האחראים על תזוזת האסטרואידים והאינטראקציות עם האובייקטים האחרים במהלך המשחק.

<u>משימה 3 – מימוש מחלקת הטורפדו:</u>

לטורפדו יש את התכונות הבאות (ייתכן ותכונות נוספות יתווספו בהמשך):

- מיקום ומהירות על ציר ה-x
- מיקום ומהירות על ציר ה-y
- כיוון התנועה (במעלות).

עליכם ליצור קובץ בשם torpedo.py ובו לממש את המחלקה Torpedo שתכיל את כל המידע הרלוונטי על הטורפדו. התכונות ישמשו אתכם בהמשך למימוש החלקים האחראים על תזוזת הטורפדואים והאינטראקציות עם האובייקטים האחרים במהלך המשחק.

<u>סיכום שלב א'</u>- בשלב זה מימשתם באופן ראשוני את המחלקות הראשיות בהם תשתמשו בתוכנית.

שלב ב'

המחלקה GameRunner, שנמצאת בקובץ ההרצה asteroids_main.py, מייצגת את המשחק הנוכחי והיא תכיל את כל המידע שיש לנו עליו - כלומר את המידע על החללית, האסטרואידים והטורפדואים. שימו לב שחלק מהקוד הדרוש כבר ממומש. באופן ספציפי, **ההתחלה** של הפונקציה __init__ כבר ממומשת, וכן מספר פונקציות נוספות שאין לשנות אותן. תוכלו להוסיף למחלקה פונקציות נוספות לשיקולכם.

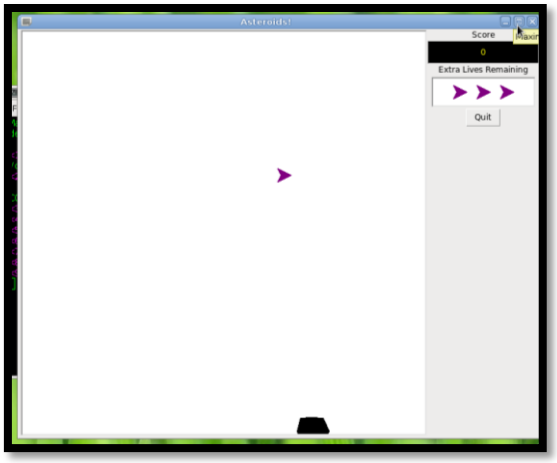
הפונקציה העיקרית של המחלקה היא הפונקציה game_loop, שלא מקבלת פרמטרים, ואותה עליכם לערוך ולעדכן לאורך השלבים הבאים בתרגיל. פונקציה זו אחראית על ביצוע הפעולות השונות במהלך המשחק, ונקראת באופן מחזורי שוב ושוב לאורך כל ריצת המשחק. על הפונקציה לבצע את כל המשימות המופיעות בשלב זה וכן בשלבים הבאים לפי הסדר (למעט מקרים בהם יצוין במפורש שהמשימה אמורה להתבצע בשלב אחר, כגון בשלב האתחול של המשחק).

שימו לב! על מנת שתוכלו לממש את המשחק יהיה עליכם לקרוא לפונקציות רבות של המחלקה **Screen**, ולכן אתם תצטרכו לעבוד בצמוד ל-**API** שלה. ז"א, עליכם לבדוק לגבי כל פונקציה של המחלקה **Screen** בה אתם מעוניינים להשתמש, מהם הפרמטרים אותם היא מקבלת ומה היא מחזירה.

<u>משימה 1 – הוספה וציור החללית:</u>

המשחק מכיל חללית יחידה, עליה השחקן שולט. עליכם להוסיף את החללית למשחק עוד בשלב אתחול המשחק כך:

1. מיקומה ההתחלתי של החללית צריך להיקבע בצורה רנדומלית בכל התחלה של המשחק. ערכי המיקום <u>יוגרלו להיות מספרים שלמים בטווח הערכים של המסך.</u>
2. מהירותה ההתחלתית של החללית (בשני הצירים) צריכה להיות 0.
3. החללית צריכה להתחיל עם כיוון 0 (אפס) - כלומר מקביל לציר ה-X (ראו תמונה מצורפת).



בשביל לצייר את החללית על המסך השתמשו בפונקציה draw_ship של המחלקה Screen, חתימת הפונקציה היא:

draw_ship(x, y, heading)

<u>משימה 2 - תזוזה של החללית:</u>

אופן התזוזה של החללית זהה למעשה לאופן התזוזה של כל אובייקט במשחק. כל אובייקט במשחק זז בכל אחד מהצירים $i \in \{x, y\}$ על פי הנוסחה הבאה:

$$NewSpot_i = ScreenMin_i + (OldSpot_i + Speed_i - ScreenMin_i)\%\Delta_i$$

כאשר:

- $ScreenMin_i$ היא הקואורדינטה המינימלית של המסך בציר ה-i.
- $ScreenMax_i$ היא הקואורדינטה המקסימלית של המסך בציר ה-i.

  (ערכים אלו נשמרו כשדות של המחלקה GameRunner בפונקציה __init__)
- $OldSpot_i$ ו-$Speed_i$ הם המיקום והמהירות הנוכחיים בציר ה-i לפני השינוי.
- $\Delta_i = ScreenMax_i - ScreenMin_i$.

שימו לב שנוסחה זו תשמש לחישוב התנועה של כלל האובייקטים במשחק (כלומר חללית, אסטרואידים, וטורפדואים).

<u>משימה 3 - שינוי כיוון החללית:</u>

את החללית ניתן לסובב בעזרת המקשים שמאלה וימינה. ניתן לדעת האם המשתמש לחץ על איזה שהוא מקש, דרך הפונקציות הרלוונטיות במחלקה Screen (הפונקציות: is_left_pressed, is_right_pressed). לחיצה על המקש שמאלה צריכה לסובב את החללית ב-7 מעלות נגד כיוון השעון (להוסיף 7 מעלות לכיוון שאליו פונה החללית של החללית), ולחיצה על המקש ימינה צריכה לסובב את החללית ב-7 מעלות עם כיוון השעון (כלומר לחסר 7 מעלות מהכיוון שאליו היא פונה).

<u>משימה 4 - האצת החללית:</u>

את החללית ניתן להאיץ בעזרת לחיצה (קצרה או ממושכת) על המקש למעלה. ניתן לדעת אם המשתמש לחץ על המקש בעזרת הפונקציה is_up_pressed של המחלקה Screen.

לחיצה על המקש למעלה צריכה להאיץ את החללית על פי הנוסחאות הבאות:

$$NewSpeed_x = OldSpeed_x + \cos(Heading)$$
$$NewSpeed_y = OldSpeed_y + \sin(Heading)$$

כאשר:

- $OldSpeed_i$ היא המהירות של החללית בציר ה-i לפני השינוי.
- $Heading$ הוא הכיוון ברדיאנים אליו פונה החללית בעת חישוב ההאצה.

שימו לב שהחללית לא מאטה מאליה, אלא רק מאיצה בכיוון שאליו מופנה החרטום. על מנת להאט את החללית יש להפנות את ראשה לצד השני ולהאיץ בכיוון זה.

<u>סיכום שלב ב'</u>- בשלב זה, בהרצה של המשחק, אמורה להיות חללית שנעה על המסך (בלי אסטרואידים), ויכולה להסתובב ולהאיץ (אך לא לירות טורפדואים).

שלב ג'

<u>משימה 1 - הוספת אסטרואידים:</u>

בפונקציה __init__, קיים פרמטר בשם asteroids_amount - הפרמטר הזה מייצג את מספר האסטרואידים בתחילת המשחק. יש להוסיף את האסטרואידים למשחק כבר בשלב ה-init.

בשביל לצייר אסטרואיד כלשהו על המסך השתמשו בפונקציה draw_asteroid שנמצאת במחלקה Screen, חתימת הפונקציה היא:

.draw_asteroid(asteroid, x, y)

עליכם לאתחל את האסטרואידים בתחילת המשחק כך:

1. קביעת מס' האסטרואידים במשחק מתבצעת ע"י שליחת מספר (מטיפוס int) כפרמטר משורת הפעלה (שורת הפקודה). אם לא נשלח שום ערך, מספר האסטרואידים הוא 5. ניתן להניח שאם נשלח פרמטר משורת הפקודה הרי שהוא מייצג מספר שלם וערכו לפחות 1.
2. המיקום הראשוני של האסטרואידים ומהירותם ייקבעו באופן רנדומלי:
   - המיקום הראשוני יוגרל כמספרים שלמים בתוך תחומי המסך. יש לוודא שהמיקום ההתחלתי של האסטרואיד לא גורם להתנגשות עם החללית (ראו פירוט בהמשך) במיקום ההתחלתי שלה, אך אין מניעה ששני אסטרואידים יהיו באותו מיקום או יתנגשו זה עם זה.
   - המהירות בכל אחד מהצירים תאותחל להיות מספר שלם, שערכו המוחלט בין 1-4, כולל.
3. הגודל ההתחלתי של אסטרואידים צריך להיות 3.
4. על מנת שאובייקט מסוג Screen 'יכיר' את האסטרואידים ויוכל להציג אותם יש להשתמש, עבור כל אסטרואיד שנוסף למשחק, בפונקציה register_asteroid שחתימתה היא:

register_asteroid(asteroid, asteroid_size)

<u>משימה 2 - הזזת האסטרואידים:</u>

עליכם לעדכן את החלק בתוכנית האחראי על תזוזת האובייקטים כך שיזיז גם את כל האסטרואידים במשחק.

<u>משימה 3 - התנגשות עם אסטרואיד:</u>

עליכם לבדוק האם אובייקטים מסוימים מתנגשים באסטרואיד (למשל טורפדו או חללית).

יש לממש במחלקה Astroid את הפונקציה has_intersection(self, obj), המקבלת כפרמטר אובייקט obj, ובודקת האם הוא התנגש עם האסטרואיד שלנו באופן הבא:

- תחילה יש לחשב את המרחק בין האובייקט לבין האסטרואיד לפי הנוסחה הבאה:

$$distance = \sqrt{(obj.x - asteroid.x)^2 + (obj.y - asteroid.y)^2}$$

- לאחר מכן יש לבדוק האם מתקיים התנאי:

$$distance \leq asteroid.radius + obj.radius$$

- במידה והתנאי מתקיים, הפונקציה צריכה להחזיר True (כלומר הייתה התנגשות) - אחרת עליה להחזיר False.

יש להוסיף שתי פונקציות (אחת במחלקה של Asteroid והשנייה במחלקה של Ship) הנותנות את רדיוס האובייקט:

1. עבור חללית - רדיוס החללית הוא 1
2. עבור אסטרואיד - רדיוס האסטרואיד נתון לפי הנוסחה: $radius = size * 10 - 5$

<u>משימה 3.1 - הפחתת חיים לחללית:</u>

בתחילת משחק עליכם לאתחל לחללית 3 יחידות "חיים". אם החללית מתנגשת עם אסטרואיד יש להוריד לה יחידת "חיים".

במקרה של התנגשות בחללית, צריכה להופיע על כך הודעת התרעה. תוכן ההודעה נתון לבחירתכם, אבל היא צריכה להיות אינפורמטיבית. בכדי להציג הודעה במשחק עליכם להשתמש בפונקציה show_message אשר שייכת למחלקה Screen וחתימתה היא:

show_message(title, message)

לשם הצגת עדכון החיים של החללית במסך השתמשו בפונקציה ()remove_life של המחלקה Screen.

מלבד הפחתת החיים, לא אמור לחול שינוי כלשהו בנתוני החללית – מיקומה, מהירותה וכיוון החרטום יישארו כפי שהיו לפני הפגיעה.

<u>משימה 3.2 - העלמת האסטרואיד שהתנגשו בו:</u>

פעולה נוספת שצריכה להתרחש לאחר התנגשות של חללית באסטרואיד היא העלמת האסטרואיד. את האסטרואיד הנפגע יש להסיר מהמסך. בכדי להסיר אסטרואיד מהמסך עליכם להשתמש בפונקציה unregister_asteroid של המחלקה Screen, המקבלת כקלט את האסטרואיד המוסר.

<u>סיכום שלב ג'</u>- בשלב זה, בהרצה של המשחק אמורה להיות חללית שנעה על המסך עם אסטרואידים. החללית יכולה להסתובב, להאיץ את מהירותה, ולהתנגש באסטרואידים (כאשר בעת התנגשות תופיע הודעה, ירדו לחללית חיים, והאסטרואיד שהחללית התנגשה בו ייעלם).

שלב ד'

משימה 1 - ציור טורפדואים:

כדי לתקוף את האסטרואידים, החללית צריכה להיות מסוגלת לירות טורפדואים. לחיצה על המקש רווח מסמנת לנו שאנו רוצים לבצע ירייה (יפורט בהמשך).

בשביל לצייר טורפדו כלשהו על המסך קיימת הפונקציה draw_torpedo שנמצאת במחלקה Screen, חתימת הפונקציה היא:

```
draw_torpedo(torpedo, x, y, heading)
```

משימה 2 - ביצוע יריה:

כאשר השחקן יילחץ על המקש רווח (ניתן לבדוק זאת על-ידי שימוש בפונקציה is_space_pressed של המחלקה Screen), עליכם להוסיף טורפדו חדש למשחק.

- המיקום ההתחלתי של טורפדו יהיה במיקום החללית בעת ששוגר.
- המהירות של הטורפדו נתונה לפי הנוסחאות הבאה:

$$TorpedoSpeed_x = ShipSpeed_x + 2 \cdot \cos(ShipHeading)$$
$$TorpedoSpeed_y = ShipSpeed_y + 2 \cdot \sin(ShipHeading)$$

  כאשר גם כאן $ShipHeading$ צריך להיות הכיוון **ברדיאנים** של הספינה בעת השיגור. מהירות הטורפדו נשארת קבועה לכל אורך חייו של הטורפדו.
- הכיוון של הטורפדו הוא הכיוון של החללית בעת השיגור, והוא נשאר קבוע לאורך כל חייו של הטורפדו.

על מנת שאובייקט מסוג Screen 'יכיר' את הטורפדואים, יש להשתמש בפונקציה register_torpedo שחתימתה היא:

```
register_torpedo(torpedo)
```

משימה 3 - הזזת טורפדואים:

עליכם לעדכן את החלק בתוכנית שלכם האחראי על תזוזת האובייקטים כך שיזיז גם את כל הטורפדואים במשחק.

משימה 4 - התנגשות טורפדו עם אסטרואיד

משימה 4.1 - בדיקת התנגשות

אתם צריכים לבדוק אם טורפדו התנגש באסטרואיד. הרדיוס של טורפדו קבוע וערכו 4 (הרדיוס הזה ניתן בשביל נוסחאות ההתנגשות).

במקרה בו טורפדו התנגש עם אסטרואיד (כלומר הפונקציה has_intersection של המחלקה Asteroid החזירה ערך True) האסטרואיד מתפוצץ אך לא נעלם. המשימות הבאות ידריכו אתכם בפעולות שצריכות להתקיים במקרה שבו התרחשה התנגשות.

<u>משימה 4.2 - הוספת נקודות למשתמש</u>

האסטרואיד מושמד ומקנה למשתמש נקודות באופן הבא:

1. אסטרואיד בגודל 3 = 20 נקודות
2. אסטרואיד בגודל 2 = 50 נקודות
3. אסטרואיד בגודל 1 = 100 נקודות

<u>הערה 1:</u> שימו לב - עליכם לשמור את הניקוד של המשתמש. בנוסף, עדכון תצוגת הנקודות על המסך מתבצע דרך המחלקה Screen באמצעות הפונקציה set_score שחתימתה היא - set_score(value).

<u>משימה 4.3 - פיצול האסטרואיד</u>

אם האסטרואיד הוא אסטרואיד גדול, כלומר גודלו גדול מ-1 האסטרואיד יתפצל לשניים בצורה הבאה: אסטרואיד בגודל 3 יהפוך לשניים בגודל 2, אסטרואיד בגודל 2 יהפוך לשניים בגודל 1, ואסטרואיד בגודל 1 ייעלם מהמסך.

שני האסטרואידים יתחילו עם אותן קואורדינטות כמו האסטרואיד הגדול יותר (לפני הפיצוץ). ההבדל בין האסטרואידים החדשים לאסטרואיד הישן הוא המהירות שלהם. חישוב המהירות החדשה, על כל ציר, נתון לפי הנוסחה הבאה:

$$\mathrm{NewAstroid}Speed_i = \pm \frac{TorpedoSpeed_i + OldAstroidSpeed_i}{\sqrt{OldAstroidSpeed_x^2 + OldAstroidSpeed_y^2}}$$

שני האסטרואידים שנוצרו צריכים לעוף במהירויות שוות אך בכיוונים מנוגדים ולכן בכל אחד מהצירים - אחד מהאסטרואידים יקבל מהירות חיובית והשני יקבל מהירות שלילית שווה בגודלה. באפשרותכם להחליט איזה אסטרואיד יקבל איזו מהירות בכל ציר, ובלבד ששניהם ינועו לאחר הפגיעה בכיוונים מנוגדים זה מזה.

את הטורפדואים שפגעו באסטרואידים יש להסיר מהמסך. בכדי להסיר טורפדו מהמסך עליכם להשתמש בפונקציה של המחלקה Screen בשם unregister_torpedo המקבלת כפרמטר את אובייקט הטורפדו המוסר.

את האסטרואיד הנפגע יש להסיר מהמסך (ולהציג רק את השניים החדשים). בכדי להסיר אסטרואיד מהמסך יש להשתמש בפונקציה של המחלקה Screen בשם unregister_asteroid המקבלת כפרמטר את אובייקט האסטרואיד המוסר.

משימה 5 - זמן חיים לטורפדו

כל טורפדו יוצא מכלל פעולה לאחר זמן מה. על מנת למדוד את זמן החיים של הטורפדו, עליכם לספור את מספר הסבבים (= מס' הקריאות לפונקציה game_loop) שבהם הוא חי, כלומר כמות הפעמים שהטורפדו זז במהלך המשחק. זמן החיים של כל טורפדו צריך להיות 200.

משימה 6 - הוספת גבול לכמות הטורפדואים של חללית

כמות הטורפדואים שיכולים להשתתף במשחק בכל רגע נתון מוגבלת ל-10, ועליכם לאכוף מגבלה זו. כלומר, אם החללית ירתה 10 טורפדואים, היא לא תוכל לירות טורפדו חדש עד שאחד מהטורפדואים הקיימים 'יצא מכלל פעולה' כלומר יעבור את זמן החיים שלו, או יתנגש באסטרואיד כלשהו.

סיכום שלב ד'- בשלב זה, בהרצה של המשחק, אמורה להיות חללית שעפה במסך עם אסטרואידים. החללית יכולה להסתובב, להאיץ את מהירותה, ולהתנגש באסטרואידים, ובנוסף החללית יכולה לירות טורפדואים.

שלב ה'

ניצחון, הפסד ויציאה מהמשחק

המשחק צריך להסתיים באחד משלושת המקרים הבאים:

(1) כל האסטרואידים במשחק התפוצצו.

(2) החללית התנגשה 3 פעמים עם אסטרואידים ולא נשארו לה יחידות חיים .

(3) נלחץ מקש היציאה 'q' (ניתן לזהות זאת בעזרת הפונקציה should_end של המחלקה Screen).

בכל אחד מהמקרים צריכה להיות מודפסת הודעה אינפורמטיבית **לבחירתכם** המסבירה את סיבת היציאה.

על מנת לסיים את המשחק ולסגור את הגרפיקה יש להשתמש בפונקציה end_game הנמצאת במחלקה Screen (פונקציה זו לא מקבלת פרמטרים). לאחר קריאה לפונקציה זו - קראו גם ל-sys.exit.

סיכום שלב ה' - בשלב זה המשחק מוכן ואתם מוזמנים לשחק בו להנאתכם :)

הערות כלליות

1. שימו לב למיקום הקבועים של התוכנית (למשל האם מספר הטורפדואים שחללית יכולה לירות צריך להישמר במחלקה של הטורפדו, של החללית או במחלקה המנהלת את המשחק?)

2. בכל פעם שעליכם להשתמש במתודה של המחלקה Screen, עליכם להפעיל אותה כמתודה של האובייקט screen__ שהוא שדה במחלקה GameRunner. את שורות הקוד האחראיות על אתחול האובייקט כבר סיפקנו לכם.
3. עבדו צמוד ל-API של המחלקה Screen. הסתכלו ובדקו מה כל פונקציה עושה, אילו פרמטרים היא צריכה לקבל ומה היא מחזירה.
4. על תרגיל זה יערכו ראיונות עם מדריכי המעבדה, בדומה לראיונות שנערכו על תרגיל 6. בראיונות תתבקשו בין היתר להסביר את השיקולים שהיו לכם בעת כתיבת המחלקות השונות והממשקים ביניהן, את ההחלטות שקיבלתם באשר לקביעת המקום בקוד בו יופיעו הפעולות השונות, וכו'.

נהלי הגשה

את תרגיל זה כאמור חייבים להגיש בזוגות.

לפני ההגשה ותחת הלינק המיועד להגשה, עליכם לפתוח קבוצה ב-moodle. אחד השותפים ייצור את הקבוצה על ידי הזנת שם השותף השני (שימו לב להוסיף את השם בדיוק כפי שהוא מופיע ב-moodle, כולל אותיות גדולות וקטנות במקומות הנכונים). השותף (שלא יצר את הקבוצה) יוכל לראות שרשמו אותו על ידי כניסה לקישור ההגשה וצפייה בשם בן הזוג. כדאי לרשום את בן הזוג בשלב מוקדם (אין צורך להגיש את התרגיל בפועל על מנת להירשם בזוג). כמו כן, וודאו כי הקבוצה אינה מכילה יותר משני שותפים.

עליכם להגיש קובץ נוסף בשם AUTHORS **(ללא כל סיומת).** קובץ זה יכיל שורה אחת ובו הלוגינים (CSE login) של שני הסטודנטים המגישים, מופרד ע"י פסיק. כך:

minniemouse,mickeymouse

ודאו כי הגשתם קובץ AUTHORS תקין! אי הגשה של קובץ AUTHORS תקין תגרור הורדה בציון.

עליכם להגיש את הקובץ ex10.zip (בלבד) בקישור ההגשה של תרגיל 10 דרך אתר הקורס על ידי לחיצה על "Upload file". אנו ממליצים להתחיל לעבוד על התרגיל בשלב מוקדם שכן התרגיל ארוך מקודמיו.

Ex10.zip צריך לכלול את הקבצים:

1. ship.py
2. asteroid.py
3. torpedo.py
4. asteroids_main.py (קובץ ההרצה הראשי אותו סיפקנו לכם).
5. AUTHORS (כפי שתואר להלן ובדומה למה שהגשתם בתרגיל 6).

קבצים נוספים הנחוצים להרצת התוכנית אם כתבתם כאלה.

בהצלחה!