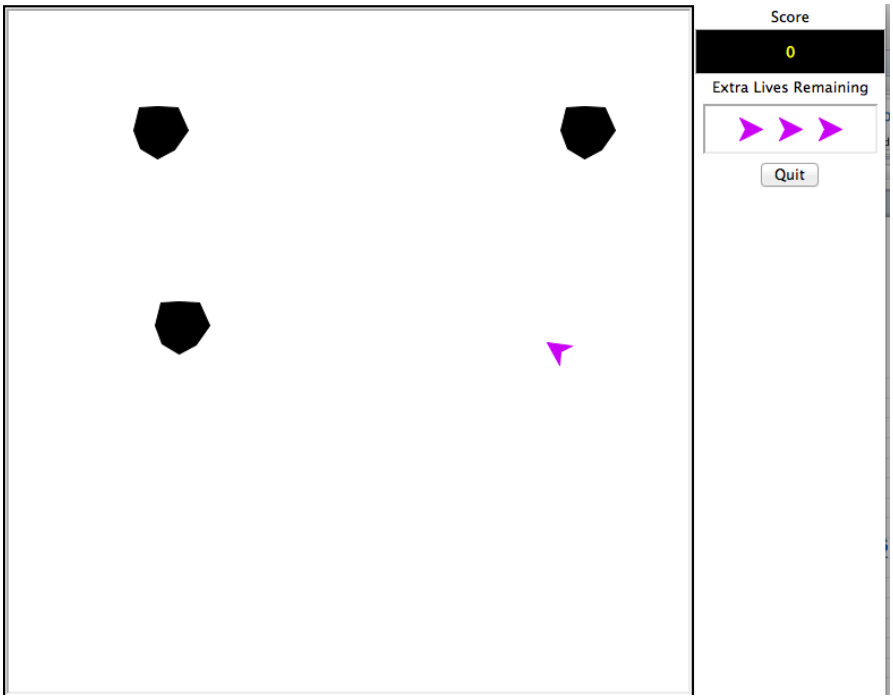
בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

מבוא למדעי המחשב 67101

תרגיל 10 - Asteroids
להגשה בתאריך 26.12.2018 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:



הערות כלליות על התרגיל:

- אפשר לפתור את התרגיל לבד (אך מומלץ לעבוד בזוגות). אם בחרתם לעבוד לבד מומלץ לשוחח על תכנון ה- OOP עם חבריכם.
- 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 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 התכנית - 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 התכנית שלכם.

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

<u>רקע מקדים:</u>

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.

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

בגישה הפסיבית: ברגע שמתבצעת פעולה ע"י המשתמש תידלק "נורה" אשר תיבדק באופן מחזורי ע"י התוכנה - ברגע שהנורה 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 בפונקצייה 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 האינטרקציות 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 <u>אתם</u> 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 האינטרקציה בין היישויות 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), שתמשיכו לעדכן 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 אינטרקציות מתקדמות בין החללית לאסטרואידים (כגון התנגשות). בסיום שלב זה המשחק שלכם יכיל אסטרואידים זזים וכן והחללית יכולה להתנגש בהם.
4. בשלב הרביעי תתבקשו לממש את החלקים האחראים על הוספה של טורפדואים, וכן אינטרקציות 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

משימה 1 – מימוש מחלקת החללית:

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

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

עליכם לממש בקובץ ship.py את המחלקה Ship שתכיל את כל התכונות הללו. התכונות ישמשו אתכם בהמשך למימוש החלקים האחראים על הזזת החללית והאינטרקציות עם האובייקטים האחרים במהלך המשחק.

משימה 2 – מימוש מחלקת האסטרואיד:

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

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

עליכם לממש בקובץ asteroid.py את המחלקה Asteroid שתכיל את כל המידע הרלוונטי על האסטרואיד. התכונות ישמשו אתכם בהמשך למימוש החלקים האחראים על תזוזת האסטרואידים והאינטרקציות 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 (במעלות).

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

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

<u>שלב ב</u>

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. הפונקציה אחראית על ביצוע הפעולות השונות האחראיות על מהלך המשחק, ונקראת באופן מחזורי שוב ושוב לאורך כל ריצת המשחק. על הפונקציה לבצע את כל המשימות המופיעות בשלב זה וכן בשלבים הבאים (למעט מקרים בהם יצויין במפורש שהמשימה אמורה להתבצע בשלב אחר, כגון בשלב ה-init)

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 ה-init של המשחק לפי ההערות המופיעות בהמשך.

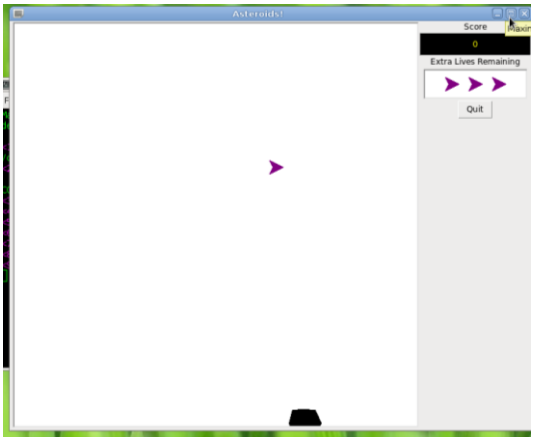
בשביל לצייר את החללית על המסך השתמשו בפונקציה draw_ship של המחלקה Screen, חתימת הפונקציה היא:

draw_ship(x, y, heading)

<u>הערה 1:</u> מיקומה ההתחלתי של החללית צריך להיקבע בצורה רנדומלית בכל התחלה של המשחק.

<u>הערה 2:</u> מהירות החללית (בשני הצירים) ההתחלתית צריכה להיות 0.

<u>הערה 3:</u> החללית צריכה להתחיל עם כיוון 0 (אפס) - כלומר מקביל לציר ה-X (ראו תמונה מצורפת).



<u>משימה 2 - תזוזה של החללית:</u>

אופן התזוזה של החללית זהה למעשה לאופן התזוזה של כל אובייקט במשחק. כל אובייקט במשחק זז באופן זהה שניתן ע"י הנוסחה הבאה:

$$NewCoord_{axis} = (Speed_{axis} + OldCoord_{axis} - AxisMinCoord)\%\Delta_{axis} + AxisMinCoord_{axis}$$

כאשר axis מייצג את התנועה על אחד מהצירים (ציר x או ציר y) והקואורדינטה המינימלית בצירים שמורה בפונקציית __init__ של המחלקה.

<u>הערה 1:</u> אתם צריכים להשתמש במהירות ומיקום האובייקט בצירים בשביל זה.

<u>הערה 2:</u> מתקיים ש- $\Delta_{axis} = AxisMaxCoord_{axis} - AxisMinCoord_{axis}$.
(ערכים אלו נשמרו כשדות של המחלקה GameRunner בפונקציה __init__).

<u>הערה 3:</u> נוסחה זו משמשת לכלל האובייקטים במשחק (כלומר חללית, אסטרואידים, וטורפדואים).

משימה 3 - שינוי כיוון החללית:

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 להזיז את החללית בכ-7 מעלות, ולחיצה על המקש ימינה צריכה להזיז את החללית בכמינוס 7 מעלות (כלומר לחסר 7 מעלות מהזווית)

הערה 1: החללית מסתובבת עם כיוון השעון, כלומר לחיצה על המקש ימינה צריכה לסובב את החללית עם כיוון השעון.

משימה 4 - האצת החללית:

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 לפי הנוסחה הבאה (עבור ציר ה-X):

$$NewSpeed_{axis} = CurrentSpeed_{axis} + \cos(CurrentHeadingInRadians)$$

עבור ציר ה-Y הנוסחה זהה, למעט החלפת פונקציית cos ב-sin.

הערה 1: המרה של הזווית (heading) של החללית ממעלות לרדיאנים מושארת כמשימה לכם.

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

שלב ג

משימה 1 - הוספת אסטרואידים:

בפונקציה __init__, קיים פרמטר בשם asteroids_amount - הפרמטר הזה מייצג את מספר האסטרואידים בתחילת המשחק. יש להוסיף את האסטרואידים 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)

הערה 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 שהמספר שנשלח כפרמטר הוא לפחות 1.

הערה 2: המיקום הראשוני של האסטרואידים ומהירותם צריכה להיות רנדומלית (המהירות תהיה בטווח של 1-4). שימו לב, יש לוודא שהמיקום ההתחלתי של האסטרואיד לא גורם להתנגשות עם החללית (ראו פירוט בהמשך) במיקום ההתחלתי שלה. הגודל ההתחלתי של אסטרואידים צריך להיות 3.

הערה 3: על מנת שאובייקט מסוג Screen 'יכיר' את האסטרואידים ויוכל להציג אותם, יש להשתמש בפונקציה register_asteroid שחתימתה היא:

```
def register_asteroid(asteroid, asteroid_size)
```

משימה 2 - הזזת האסטרואידים:

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

משימה 3 - התנגשות עם אסטרואיד:

עליכם לבדוק האם אובייקטים מסוימים מתנגשים באסטרואיד (למשל טורפדו או חללית). על כן יש לממש את הפונקציה הבאה במחלקה Asteroid:

```
has_intersection(self, obj)
```

הפונקציה תבדוק האם האובייקט obj התנגש עם האסטרואיד שלנו באופן הבא: תחילה יש לחשב את המרחק בין האובייקט לבין האסטרואיד לפי הנוסחה הבאה (הסבר לשימוש בנוסחה זו):

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

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.

כעת, 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:

```
size * 10 - 5
```

הערה 1: המספר 10 הוא מקדם הגודל, והמספר "מינוס 5" מייצג גורם נרמול.

משימה 3.1 - הפחתת חיים לחללית:

אם החללית מתנגשת עם אסטרואיד יש להוריד לה "חיים", לחללית יש 3 חיים בתחילת המשחק.

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)

<u>הערה 1:</u> לשם עדכון החיים של החללית במסך השתמשו בפונקציה ()remove_life של המחלקה Screen.

<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 ב-unregister_asteroid המקבלת 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).

<u>שלב ד</u>

<u>משימה 1 - ציור טורפדואים:</u>

כדי לתקוף את האסטרואידים, על החללית להיות בעלת יכולת 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)

<u>משימה 2 - ביצוע יריה:</u>

כאשר השחקן יילחץ על המקש רווח (ניתן לבדוק זאת על-ידי שימוש בפונקציה is_space_pressed של המחלקה Screen), עליכם להוסיף טורפדו חדש למשחק. המהירות של הטורפדו עבור ציר ה-X תינתן לפי הנוסחה הבאה:

$$NewSpeed_{axis} = CurrentSpeed_{axis} + 2 \cdot \cos(CurrentHeadingInRadians)$$

<u>הערה 1:</u> עבור ציר ה-Y הנוסחה זהה, למעט cos שמשתנה ל-sin.

<u>הערה 2:</u> על מנת שאובייקט מסוג Screen 'יכיר' את הטורפדואים, יש להשתמש בפונקציה register_torpedo שחתימתה היא:

def register_torpedo(torpedo)

הערה 3: הכיוון והמיקום ההתחלתי של הטורפדו צריך להיות הכיוון של החללית בזמן הירייה.

הערה 4: מהירות הטורפדו קבועה ולא משתנה לכל אורך חייו.

הערה 5: המספר 2 בנוסחה הנ"ל הוא "גורם תאוצה".

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.

משימה 4.2 - הוספת נקודות למשתמש

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

הערה 1: שימו לב - עליכם לשמור את הניקוד של המשתמש. עדכון הנקודות על המסך מתבצע דרך המחלקה Screen באמצעות הפונקציה set_score שחתימתה היא:

```
set_score(value)
```

משימה 4.3 - פיצול האסטרואיד

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:

$$NewSpeed_{axis} = \frac{TorpedoSpeed_{axis} + CurrentSpeed_{axis}}{\sqrt{CurrentSpeed_x^2 + CurrentSpeed_y^2}}$$

<u>הערה 1:</u> עליכם ליצור תנועה בביוונים נגדיים של האסטרואידים החדשים, כלומר עבור אחד האסטרואידים החדשים, הכפילו את המהירות המקורית (על שני הצירים) ב- 1-.

<u>הערה 2:</u> את הטורפדואים שפגעו באסטרואידים יש להסיר מהמסך. בכדי להסיר טורפדו מהמסך עליכם להשתמש ב-unregister_torpedo המקבלת את אובייקט הטורפדו המוסר.

<u>הערה 3:</u> את האסטרואיד הנפגע יש להסיר מהמסך (ולהציג רק את השניים החדשים). בכדי להסיר אסטרואיד מהמסך יש להשתמש ב-unregister_asteroid המקבלת את אובייקט האסטרואיד המוסר.

<u>משימה 5 - זמן חיים לטורפדו</u>

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.

<u>משימה 6 - הוספת גבול לכמות הטורפדואים של חללית</u>

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.

<u>סיכום שלב ד'</u> - 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.

<u>שלב ה'</u>

<u>משימה 1 - ניצחון, הפסד ויציאה מהמשחק</u>

המשחק צריך להסתיים באחד משלושת המקרים הבאים: (1) כל האסטרואידים במשחק התפוצצו (2) לא נותרו חיים לחללית (מספר החיים שלה הגיע ל-0) (3) נלחץ מקש היציאה 'q' (ניתן לזהות זאת בעזרת הפונקציה should_end של המחלקה Screen).

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

<u>הערה 1:</u> תוכן ההודעה לא מוכתב לכם ובלבד שיהיה אינפורמטיבי.

הערה 2: 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.

משימה 2 – שיגור החללית

השחקן יכול לבחור לשגר את החללית שלו למיקום אקראי אחר על המסך על-ידי לחיצה על המקש 't' (ניתן לדעת האם לחץ בעזרת הפונקציה is_teleport_pressed של המחלקה Screen).

הערה 1: לאחר הבחירה בשיגור החללית, יש להעביר את מיקום החללית למיקום אחר אקראי על המסך, אך יש לוודא כי המיקום פנוי ואין התנגשות של החללית עם אף אסטרואיד.

הערה 2: כיוון ומהירות החללית (בשני הצירים) צריכים להיות זהים לאלו שהיו לחללית לפני השיגור.

משימה 3 – ירייה מיוחדת (בונוס עד 5 נקודות):

השחקן יכול לבחור לירות ירייה מיוחדת על-ידי לחיצה על המקש 's' (ניתן לזהות זאת בעזרת הפונקציה is_special_pressed של המחלקה Screen). את אופי הירייה המיוחדת אנו משאירים לכם.

הערה 1: לכל ירייה מיוחדת יש זמן חיים המוגבל ל-150 סיבובים של המשחק.

הערה 2: בכל רגע נתון יכולים להשתתף במשחק 5 "יריות מיוחדות" בלבד, ועליכם לאכוף מגבלה זו.

הערה 3: זוהי משימת בונוס, ונקודות ינתנו על תקינות הקוד, וכן יצירתיות של אופי הירייה המיוחדת.

הערה 4: יש לתאר בקובץ ה-README את אופי הירייה המיוחדת שבחרתם.

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

חלק ו' - כתיבת קובץ **README**

אתם מתבקשים לכתוב 3 שיקולים שהיו לכם בעיצוב המשחק, כלומר 3 החלטות שהייתם צריכים להחליט בנוגע לעיצוב התכנית. לגבי כל אחת מההחלטות הללו, עליכם לרשום אלטרנטיבה חלופית, לציין יתרון של האפשרות אותה לא בחרתם ולציין את הסיבה שבגינה בחרתם באפשרות שלכם.

בנוסף, במידה ובחרתם לבצע את משימת הבונוס, יש לתאר את אופי הירייה מיוחדת שבחרתם.

הערות כלליות

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

2. בכל פעם שעליכם להשתמש במתודה של המחלקה Screen, עליכם להפעיל אותה דרך האוביקט screen__שהוא שדה במחלקה GameRunner. את שורות הקוד האחראיות על אתחול האוביקט כבר סיפקנו לכם.
3. עבדו צמוד ל-API של המחלקה Screen. הסתכלו ובידקו מה כל פונקציה עושה, אילו פרמטרים היא צריכה לקבל ומה היא מחזירה.

פתרון בית הספר

ממומשת עבורכם גרסה של המשחק. מומלץ לאחר קריאת התרגיל וטרם תחילת המימוש להריץ את המשחק.
המשחק אינטואיטבי ומספר משחקים בו מבהירים את הכוונה הכללית על ניהול המשחק כמו גם שאלות על פרטים ספציפיים.
הפיתרון הוא הקובץ המקופמל asteroids_main הנמצא בתיקייה :

~intro2cs1/bin/ex10/

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

~intro2cs1/bin/ex10/asteroids <num_asteroids>

כאשר הפרמטר num_asteroids הינו אופציונאלי

נהלי הגשה

את תרגיל זה אנו ממליצים לבצע בזוגות. על כל זוג להגיש את התרגיל מלוגין אחד בלבד !

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

minniemouse,mickeymouse

במידה ובחרתם להכין ולהגיש את התרגיל לבד, עליכם להוסיף את קובץ הAUTHORS עם הלוגין שלכם בלבד (ללא פסיק).

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. README (כפי שמפורט בקובץ נהלי הקורס)
6. AUTHORS (כפי שתואר להלן ובדומה למה שהגשתם בתרגיל 6).

מותר לכם לצרף קבצים נוספים, אך אנא דאגו לתעד אותם ב-README.

בהצלחה !