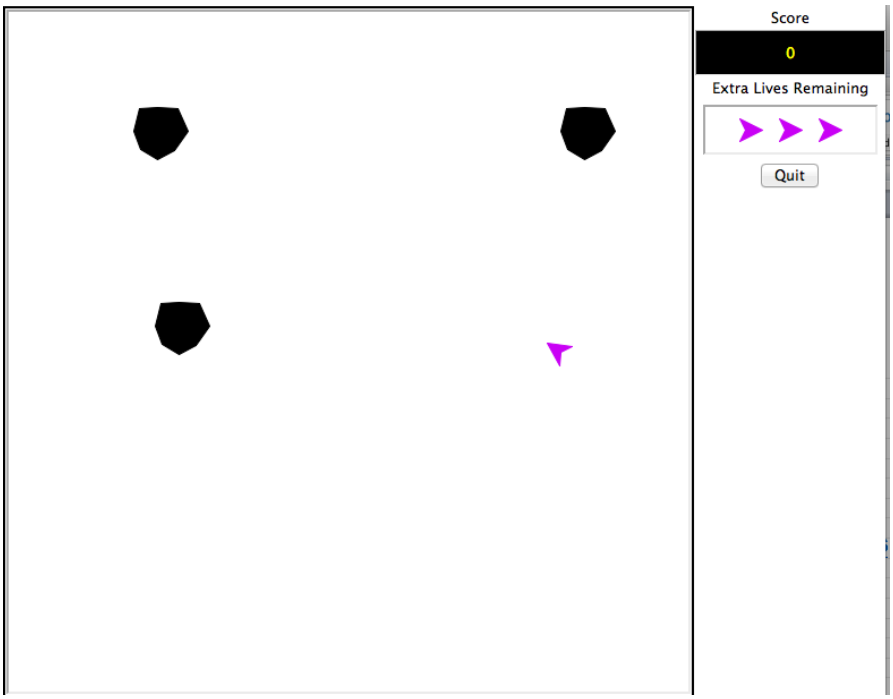
בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

מבוא למדעי המחשב 67101

תרגיל 9 - Asteroids
להגשה בתאריך 11/01/2017 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.
- המטרה של התרגיל היא התנסות בתכנון מחלקות ועיצוב כללי של מערכת. התרגיל צריך להראות כמו פתרון בית הספר, על כן אם תבחרו לממש את התרגיל בצורה שונה ממה שמתואר למטה אך ההתנהגות תהיה דומה לפתרון בית הספר - הפתרון יתקבל.

- הרבה פונקציות שעליכם לממש נמצאות בקובץ asteroids_main.py, בנוסף לכך, ניתן 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 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, ותכנות 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 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 (כמתואר למטה). את רצף הפעולות הזה תצטרכו לממש בפונקצייה `game_loop` לפי סדר מסוים.

הפונקציה `game_loop` נתונה לכם בקובץ asteroids_main.py, את הפונקציה הזו <u>אתם</u> תצטרכו להשלים לפי המשימות בשלבים א'-ה' (בסדר הזה) הנתונים למטה. מכיוון שמימוש כלל המשימות (מלבד הראשונה שהינה פונקציית עזר) יחרוג מהאורך המותר של פונקציה בקורס, אנחנו משאירים לכם להפעיל שיקול דעת ולכתוב פונקציות נוספות שייעזרו לכם בפתרון כל משימה – שימו לב! יכול להיות שאותה פונקציה תוכל להועיל לכם ביותר ממשימה אחת.

המטרה של התרגיל הינה כתיבת אובייקטים מורכבים וחשיבה על האינטרקציה בין היישויות השונות בתוכנית.

<u>מבנה התרגיל:</u>

1. בשלב הראשון תתבקשו לממש את האובייקטים המרכזיים במשחק שהם החללית והאסטרואידים.
2. בשלב השני עליכם לממש את פונקציית המשחק הראשית (`game_loop`). את הפונקציה הזאת תמשיכו לעדכן במהלך השלבים הבאים.
3. בשלב השלישי תתבקשו לממש אינטרקציות מתקדמות בין החללית לאסטרואידים (כגון התנגשות) לממש את הטורפדו הראשון. בשלב זה המשחק אמור לעבוד כמצופה.
4. בשלב הרביעי (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 - חללית (תכונות נוספות ייתווספו בהמשך):</u>

לחללית יש את התכונות הבאות:

- מיקום ומהירות על ציר ה-x
- מיקום ומהירות על ציר ה- y
- כיוון (במעלות). החללית צריכה להתחיל עם כיוון 0 (אפס) - כלומר במקביל לציר ה-X בכיוון החיובי (ראו איור)

עליכם לממש בקובץ ship.py את המחלקה Ship שתכיל את כל התכונות הללו, אתם תהיו אחראיים על הזזת החללית במהלך המשחק. בשביל לצייר את החללית על המסך השתמשו בפונקציה `draw_ship` שנמצאת באובייקט 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>: החללית מסתובבת עם כיוון השעון, כלומר לחיצה על המקש ימינה צריכה לסובב את החללית עם כיוון השעון.

<u>משימה 2 - אסטרואיד (תכונות נוספות ייתווספו בהמשך):</u>

לאסטרואיד שלנו יש את התכונות הבאות:

- 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. כמו כן, אתם תהיו אחראיים על תזוזת האסטרואידים במהלך המשחק. בשביל לצייר אסטרואיד כלשהו על המסך השתמשו בפונקציה draw_asteroid שנמצאת באובייקט Screen, חתימת הפונקציה היא:

```
draw_asteroid(asteroid,x,y)
```

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

<u>שלב ב</u>

<u>מימוש הפונקציה **game_loop** (תכונות נוספות ייתווספו בהמשך):</u>

הפונקציה הנ"ל נמצאת בתוך המחלקה GameRunner שנמצאת בקובץ ההרצה asteroids_main.py. המחלקה 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.

הפונקציה הראשית שאיתה תעבדו היא הפונקציה game_loop_ שלא מקבלת פרמטרים.

<u>משימה 1 - תזוזה של החללית</u>

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- $\Delta_{axis} = AxisMaxCoord_{axis} - AxisMinCoord_{axis}.$

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

<u>משימה 2 - שינוי כיוון החללית</u>

את החללית ניתן לסובב בעזרת המקשים שמאלה וימינה. ניתן לדעת האם המשתמש לחץ על איזה שהוא מקש, דרך הפונקציות הרלוונטיות באובייקט 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 לחסר7 מעלות מהזווית(

<u>הערה 1:</u> כן, 7 ו-"מינוס 7" נחשבים כ-Magic Numbers.

<u>הערה 2:</u> אין חשיבות לסדר הפעולות שבהן החללית זזה.

משימה 3 - האצת החללית

את החללית ניתן להאיץ בעזרת לחיצה (קצרה או ממושכת) על המקש למעלה. ניתן לדעת אם המשתמש לחץ על מקש כלשהו על ידי הפונקציה הרלוונטיות באובייקט Screen (שהיא `is_up_pressed` במקרה זה).

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 היא:

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

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

סיכום שלב ב' - בשלב זה, בהרצה של המשחק, אמורה להיות חללית שעפה במסך (בלי אסטרואידים), ויכולה להסתובב ולהאיץ.

שלב ג

משימה 1 - הוספת אסטרואידים

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

הערה 1: ניתן לשנות את כמות האסטרואידים במשחק ע"י שליחת מספר (מטיפוס int) כפרמטר משורת הפעלה. ערך ברירת המחדל של מספר האסטרואידים הוא 3 (כלומר, אם לא נשלח פרמטר בשורת הפעלה, המספר ההתחלתי של אסטרואידים הוא 3). ניתן להניח שהמספר שנשלח כפרמטר הוא לפחות 1.

הערה 2: המיקום הראשוני של האסטרואידים ומהירותם צריכה להיות רנדומלית (אך יש לוודא שהמיקום ההתחלתי שונה מזה של החללית). הגודל ההתחלתי של אסטרואידים צריך להיות 3.

הערה 3: על מנת שהאובייקט Screen 'יכיר' את האסטרואידים, יש להשתמש בפונקציה `register_asteroid` שחתימתה היא:

```
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

<u>הערה 1:</u> המספרים 10 ו-"מינוס 5" הם Magic Numbers , המספר 10 הוא מקדם הגודל, והמספר "מינוס 5" מייצג גורם נרמול.

<u>משימה 3.1 - הפחתת חיים לחללית</u>

אם החללית מתנגשת עם אסטרואיד יש להוריד לה "חיים", החללית צריכה להתחיל עם 3 חיים בהתחלת המשחק.

במקרה 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 `screen.remove_life()`

<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 שעפה במסך עם אסטרואידים. החללית יכולה להסתובב, להאיץ את מהירותה, ולהתנגש באסטרואידים (כאשר בעת התנגשות תופיע הודעה והאסטרואיד שהתנגשנו בו ייעלם).

<u>שלב ד</u>

<u>משימה 1 - הוספת טורפדואים</u>

כדי לתקוף את האסטרואידים, על החללית להיות בעלת יכולת לירות טורפדואים. לחיצה על המקש רווח מסמנת לנו שאנו רוצים לבצע ירייה. לשם ביצוע שלב זה הגדירו בתוך הקובץ torpedo.py את המחלקה Torpedo אשר תכיל:

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

<u>הערה 1:</u> בשביל להוסיף את הטורפדו למשחק עליכם להשתמש בפונקציה `register_torpedo` שחתימתה היא:

```
def register_torpedo(torpedo)
```

<u>הערה 2:</u> הכיוון והמיקום ההתחלתי של הטורפדו צריך להיות הכיוון של החללית בזמן הירייה.

<u>הערה 3:</u> שימו לב שגם כאן אתם תהיו אחראיים על הזזת הטורפדואים. בשביל לצייר טורפדו כלשהו על המסך קיימת הפונקציה draw_torpedo שנמצאת באובייקט Screen, חתימת הפונקציה היא:

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

<u>משימה 2 - ביצוע יריה</u>

כאשר השחקן יילחץ על המקש רווח, עליכם להוסיף טורפדו חדש למשחק. המהירות של הטורפדו עבור ציר ה-X תינתן לפי הנוסחה הבאה:

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

המהירות של הטורפדו עבור ציר ה- Y תינתן לפי הנוסחה הבאה:

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

<u>הערה :1</u> מהירות הטורפדו קבועה ולא משתנה לכל אורך חייו.

<u>הערה:</u> המס' 2 בנוסחה הנ"ל הוא "גורם תאוצה" וגם הוא magic number.

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 `get_intersection` החזיר 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 (כלומר צריכה להיות דרך לקבל את גודל האסטרואיד - רמז מתודה חדשה), האסטרואיד יתפצל לשניים בצורה הבאה:

שני האסטרואידים יתחילו עם אותן קואורדינטות כמו האסטרואיד הגדול יותר (לפני הפיצוץ) ובגודל קטן יותר (חשוב! אסטרואיד בגודל 3 יהפוך לשניים בגודל 2, ואסטרואיד בגודל 1 ייעלם מהמסך), ההבדל בין האסטרואידים החדשים לאסטרואיד הישן הוא המהירות שלהם. חישוב המהירות החדשה, על כל ציר, נתון לפי הנוסחה הבאה (כאשר CurrentSpeed מתייחס למהירות האסטרואיד הגדול לפני שהתחלק):

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

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

הערה 2: את הטורפדואים שפגעו באסטרואידים יש להסיר מהמסך.

הערה 3: את האסטרואיד הנפגע יש להסיר מהמסך (ולהציג רק את השניים החדשים). בכדי להסיר אסטרואיד מהמסך יש להשתמש ב-`unregister_asteroid` המקבלת את ה-id של 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 שבהם הוא חי, כלומר כמות הפעמים שהטורפדו זז במהלך המשחק. זמן החיים של כל טורפדו צריך להיות 200. בכדי להסיר טורפדו מהמסך עליכם להשתמש ב-`unregister_torpedo` המקבלת את הטורפדו המוסר.

משימה 6 - הוספת גבול לכמות הטורפדואים של חללית

כמות הטורפדואים שיכולים להשתתף במשחק בכל רגע נתון מוגבלת ל-15, ועליכם לאכוף מגבלה זו. כלומר, אם החללית ירתה 15 טורפדואים, היא לא תוכל לירות טורפדו חדש עד שאחד מהטורפדואים הקיימים 'יצא מכלל פעולה' כלומר יעבור את זמן החיים שלו.

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'

משימה 1 - ניצחון, הפסד ויציאה מהמשחק

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'.
בכל אחד מהמקרים צריכה להיות מודפסת הודעה המסמנת את סיבת היציאה.

הערה 1: 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`.

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.

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?)

שאלות ופניות

שימו לב! כל שאלה הקשורה לתרגיל יש לשאול בפורום המיועד לתרגיל זה, הנמצא באתר הקורס:

http://www.cs.huji.ac.il/~intro2cs

בקשות אישיות בלבד (כמו בקשה לדחייה במועד ההגשה) יש להפנות למייל הקורס: intro2cs@cs.huji.ac.il, על פי ההוראות המפורטות בקובץ נהלי הקורס.

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:

~intro2cs/bin/ex9/

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

~intro2cs/bin/ex9/asteroids <num_asteroids>

כאשר הפרמטר num_asteroids הינו אופציונאלי

נהלי הגשה

יצירת קובץ **zip**

בתרגיל זה התבקשתם ליצור ולעדכן את הקבצים הבאים:

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 שמפורט בקובץ נהלי הקורס ובדומה למה שהגשתם בתרגיל 5).

<u>הערה</u> מותר לכם לצרף קבצים נוספים, אנא דאגו לצרף אותם לתרגיל ולתעד אותם ב-README

כעת עליכם ליצור קובץ zip הנקרא ex9.zip המכיל את חמשת הקבצים הנ"ל וכל קובץ אחר שייצרתם בשביל להריץ את התוכנית (אין צורך בקבצים נוספים אבל אתם לא מוגבלים ל5 הקבצים האלו).

בווינדוס בחרו את הקבצים ולחצו מקש ימני, לאחר מכן בחרו ב-send to ובחרו באפשרות של "Compressed (zipped) folder".

בלינוקס ניתן לעשות זאת בעזרת פקודת ה-shell הבאה (כאשר אתם נמצאים בתיקייה 9ex שיצרתם):

```
zip ex9.zip ship.py asteroid.py torpedo.py
asteroids_main.py README AUTHORS
```

(ראו במצגת של התרגול הראשון הסבר לגבי קבצי zip.)

- זכרו את האזהרה מהתרגול הראשון – אם אתם שוכחים לכתוב את שם קובץ ה-zip שאתם רוצים ליצור, אתם תדרסו ותהרסו את הקובץ הראשון שאתם כותבים בפקודה הנ"ל, וקובץ זה ישתנה ויהפוך להיות קובץ zip המכיל את הקבצים האחרים. למשל אם תכתבו את הפקודה:
  `zip ship.py asteroid.py torpedo.py asteroids_main.py README AUTHORS`
  הקובץ ship.py שכתבתם יידרס!

- מומלץ לבדוק את קובץ ה-zip שיצרתם על ידי העתקת התוכן שלו לתיקייה נפרדת ופתיחתו (extract) בעזרת ביצוע הפקודה: **`unzip ex9.zip`**, ולאחר מכן יש לבדוק באמצעות הפקודה **`ls -h`** שכל הקבצים הדרושים קיימים שם ולא ריקים.

סקריפט קדם-הגשה (**Pre submit script**): זהו סקריפט לבדיקה בסיסית של קבצי ההגשה של התרגיל. על מנת להריץ את הסקריפט לתרגיל 1 <u>יש להשתמש במחשבי בית הספר</u> (או פיסית או כאשר מתחברים מרחוק) הקלידו את הפקודה הבאה בתיקיה בה נמצא הקובץ ex1.zip שייצרתם:

~intro2cs/bin/presubmit/ex9 ex9.zip

הסקריפט מייצר הודעת הצלחה במקרה של מעבר כל הבדיקות הבסיסיות והודעות שגיאה רלוונטיות במקרה של כישלון בחלק מהבדיקות.

שימו לב, סקריפט קדם ההגשה נועד לוודא רק תקינות בסיסית ביותר ומעבר של בדיקות הסקריפט לא מבטיח את תקינותה של התוכנית! עליכם לוודא בעצמכם שהתוכנית שלכם פועלת כפי שדרוש.

הגשת קובץ **zip**

עליכם להגיש את הקובץ ex9.zip בקישור ההגשה של תרגיל 9, על ידי לחיצה על "Upload File".

שימו לב שהגשת תרגיל דורשת שתהיו מחוברים עם ה-user והסיסמא שלכם (שנרשמתם איתם למערכת CS).

הנכם רשאים להגיש תרגילים דרך מערכת ההגשות באתר הקורס מספר רב של פעמים. ההגשה האחרונה בלבד היא זו שקובעת ושתיבדק.

לאחר הגשת התרגיל, ניתן ומומלץ להוריד אותו ולוודא כי הקבצים המוגשים הם אלו שהתכוונתם להגיש וכי הקוד עובד על פי ציפיותיכם.

קראו היטב את קובץ נהלי הקורס לגבי הנחיות נוספות להגשת התרגילים.

שימו לב - יש להגיש את התרגילים בזמן!

בהצלחה !