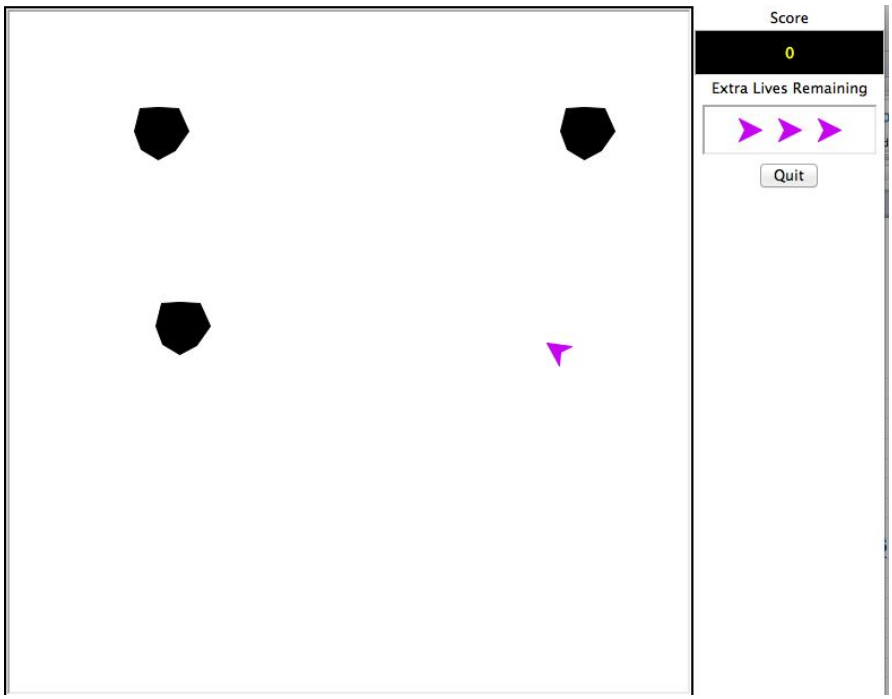
מבוא למדעי המחשב 67101

תרגיל 9 - Asteroids
להגשה בתאריך 23/11/2015 בשעה 22:00

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

בסוף התרגיל אתם תייצרו משחק שייראה ככה:



הערות כלליות על התרגיל:

- מותר לפתור את התרגיל לבד (אך מומלץ לעבוד בזוגות). אם בחרתם לעבוד לבד אנחנו ממליצים לכם לשוחח על תכנון ה- OOP עם חבריכם.
- התרגיל הינו ארוך, התחילו לעבוד עליו מוקדם!
- ביצוע המשימות לפי סדר יבטיח לכם פעולה נכונה של המשחק, אך ישנן הרבה דרכים שניתן לבצע זאת - המשימות למטה הינן הדרכה בלבד.
- המטרה של התרגיל הינה שתתנסו בתכנון מחלקות ועיצוב כללי של מערכת. התרגיל צריך להראות כמו הפתרון בית ספר, על כן אם תבחרו לממש את התרגיל בצורה שונה ממה שמתואר למטה (אך ההתנהגות תשאר זהה) - הפתרון ייתקבל.
- ישנן הרבה פונקציות שעליכם לממש בקובץ asteroids_main.py, בנוסף מותר לכם להוסיף קבצים משלכם (מעבר לקבצים שאנחנו מבקשים מכם)

לצורך מימוש התרגיל מימשנו עבורכם את המחלקה Screen הנמצאת בקובץ screen.py, מחלקה זו אחראית על ציור האובייקטים למסך ואתם מחוייבים 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 שימוש בה בכל התוכנית.

בשביל 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 התעכבות.

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 בשנייה שהיא מבוצעת.

אנחנו בתרגיל ננקוט בגישה הפסיבית, מכיוון שאנחנו מייצרים משחק ניתן להתייחס לכל הפעולות כאילו הן קורות ברצף (לפי המתואר למטה) קבוע כלשהו, את רצף הפעולות הזה תצטרכו לממש בפונקצייה game_loop לפי סדר מסויים.

הפונקצייה 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 שייתנו לכם (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 מרוכבים 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 תצטרכו לממש את פונקציית המשחק הראשית (game_loop), אתם תמשיכו לעדכן את הפונקציה הנ״ל במשך השלבים הבאים גם כן.**
3. **בשלב השלישי תתבקשו לממש אינטרקציות מתקדמות בין החללית לאסטרואידים (התנגשות למשל) ומימוש הטורפדו הראשון שלכם, אחרי שלב זה המשחק צריך לעבוד כמצופה.**
4. **בשלב הרביעי (והאחרון) תתבקשו לממש אלמנטים מתקדמים במשחק.**

המלצת צוות הקורס הינה שתעקבו אחר המשימות כמו שהן ותבדקו בכל שלב שאתם אכן עומדים בדרישות המתאימות.

שלב א

משימה 1 - חללית (תכונות נוספות ייתווספו בהמשך):

לחללית שלנו 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 מקביל לציר ה-X.**

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 ישנה הפונקציה draw_ship שנמצאת באובייקט Screen, חתימת הפונקציה הינה:

```
draw_ship(x,y, heading)
```

הערה 1: שימו לב שהגדרנו עבורכם פונקציות (ריקות) לציור האובייקטים במחלקה GameRunner, כלומר אתם יכולים להכניס את הקוד שאחראי על ציור החללית (ושאר האובייקטים) בפונקציות המתאימות.

הערה 2: מיקומה ההתחלתי של החללית צריך להקבע בצורה רנדומית בכל התחלה של המשחק.

הערה 3: כיוון הסיבוב של החללית הוא עם כיוון השעון, כלומר לחיצה על המקש ימינה צריכה לסובב אותו עם כיוון השעון.

משימה 2 - אסטרואיד (תכונות נוספות ייתווספו בהמשך):

לאסטרואיד שלנו יש את התכונות הבאות:

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

עליכם לממש בקובץ asteroid.py את המחלקה Asteroid שתכיל את כל המידע הרלוונטי על האסטרואיד, אתם תהיו אחראיים על הזזת האסטרואידים במהלך המשחק. בשביל לצייר אסטרואיד כלשהו על המסך ישנה הפונקציה draw_asteroid שנמצאת באובייקט Screen, חתימת הפונקציה הינה:

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

סיכום שלב א' - כרגע אתם ייצרתם את המחלקות הראשיות בהם תשתמשו בתוכנית.

שלב ב

משימה 1 - מימוש הפונקציה game_loop (תכונות נוספות ייתווספו בהמשך):

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, אסטרואידים וטורפדואים. שימו לב שכבר מימשנו עבורכם חלק מהקוד הדרוש, ספציפית התחלנו לממש עבורכם את הפונקציה __init__ ועוד מספר פונקציות שאסור לכם להתעסק איתם.

הפונקציה הראשית שאיתה תעבדו הינה הפונקציה game_loop_ אשר אינה מקבלת פרמטרים.

משימה 1 - תזוזה של החללית

כל (כן, כל) אובייקט במשחק שלנו זז באותו האופן שניתן ע"י הנוסחא הבאה:

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

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

הערה 1: אתם צריכים להשתמש במהירות ומיקום האובייקט בצירים בשביל זה.

הערה 2: מתקיים ש- $\Delta_{axis} = AxisMaxCoord_{axis} - AxisMinCoord_{axis}$.

הערה 3: אנחנו נשתמש בנוסחא זו לכלל האובייקטים במשחק (כלומר גם חללית גם אסטרואידים וגם טורפדואים).

משימה 2 - שינוי כיוון החללית

את החללית ניתן לסובב בעזרת המקשים שמאלה וימינה, אתם יכולים לדעת האם המשתמש לחץ על איזה שהוא מקש דרך הפונקציות הרלוונטיות באובייקט Screen שלכם ע"י הפונקציות:,is_left_pressed is_right_pressed. לחיצה על המקש שמאלה צריכה להזיז את החללית בכ-7 מעלות לעומת זאת לחיצה על המקש ימינה צריכה להזיז את החללית בכ-"מינוס 7 מעלות (כלומר בערך 7-)

הערה 1: כן 7 ו-"מינוס 7" נחשבים כ-Magic Numbers.

הערה 2: אין חשיבות לסדר הפעולות שבהן החללית זזה.

משימה 3 - האצת החללית

את החללית ניתן להאיץ בעזרת לחיצה (קצרה או מתמשכת) על המקש למעלה, אתם יכולים לדעת האם המשתמש לחץ על איזה שהוא מקש דרך הפונקציה הרלוונטיות באובייקט Screen שהיא is_up_pressed

לחיצה על המקש למעלה צריכה להאיץ את החללית לפי הנוסחאה הבאה (עבור ציר ה-X):

כאשר על ציר -Y אנחנו נשתמש בנוסחא זהה רק שבמקום cos אנחנו נשתמש ב- sin.

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

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

סיכום שלב ב' - כרגע אם תריצו את המשחק צריכה להיות לכם חללית שעפה במסך בלי שום אסטרואידים, **החללית יכולה להסתובב ולהאיץ את מהירותה.**

שלב ג

משימה 1 - הוספת אסטרואידים

אם תשימו לב, בפונקציה __init__ אתם מקבלים פרמטר בשם asteroids_amnt - הפרמטר הזה מייצג את מספר האסטרואידים שיש להתחיל איתם במהלך המשחק. עליכם להוסיף את האסטרואידים כבר בשלב ה-init.

הערה 1: אתם יכולים לשנות את כמות האסטרואידים במשחק ע"י שליחת מספר (מטיפוס int) כפרמטר משורת הפעלה, אם לא תשלחו כלום מספר האסטרואידים יהיה 3. אתם יכולים להניח שבבדיקות שלנו (ולכן גם אצלכם) המספרים שנבחר יהיו תמיד חיוביים וגדולים מ-1.

הערה 2: מיקום האסטרואידים הראשונים ומהירותם ההתחלתית צריכה להיות רנדומית (אך שלא תהיה זהה לזו של החללית). גודלם הראשוני צריך להיות 3.

הערה 3: בשביל שהאובייקט Screen יכיר את האסטרואידים שאתם יוצרים עליכם להשתמש בפונקציה register_asteroid שחתימתה היא:

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

משימה 2 - הזזת האסטרואידים

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

משימה 3 - התנגשות עם אסטרואיד

עליכם לבדוק האם אובייקטים מסויימים מתנגשים באסטרואיד (למשל טורפדו או חללית), על כן עליכם להוסיף את הפונקציה הבאה אל המחלקה Asteroid:

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

הפונקציה תבדוק האם האובייקט obj התנגש עם האסטרואיד שלנו לפי הרעיון הבא: תחילה נחשב את המרחק בין האובייקט לבין האסטרואיד לפי הנוסחה הבאה (הסבר למדוע אנחנו משתמשים בנוסחה זו):

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

לאחר מכן נבדוק האם מתקיים התנאי:

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

אם מתקיים נחזיר True (כלומר הייתה התנגשות) - אחרת נחזיר False.

עכשיו, נצטרך להוסיף שתי פונקציות (אחת במחלקה של Asteroid והשנייה במחלקה של Ship) הנותנות לנו את רדיוס האובייקט:

1. עבור חללית - רדיוס החללית הינו 1
2. עבור אסטרואיד - רדיוס האסטרואיד יינתן לפי הנוסחאה:

size * 10 - 5

<u>הערה 1:</u> המספרים 10 ו-"מינוס 5" <u>הינם</u> Magic Numbers, המספר 10 הינו מקדם הגודל והמספר "מינוס 5" **מייצג גורם נירמול.**

<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 בו התנגשנו בחללית צריכה להופיעה הודעה אשר מתריעה על כך שהתנגשנו באסטרואיד, תוכן ההודעה נתון לבחירתכם אבל צריך להיות אינפורמטיבי. בכדי להציג הודעה במשחק עליכם להשתמש בפונקציה show_message אשר שייכת למחלקה Screen וחתימתה הינה:

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 הינה 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> כרגע אם תריצו את המשחק צריכה להיות לכם חללית שעפה במסך עם אסטרואידים, החללית יכולה להסתובב ולהאיץ את מהירותה ובנוסף להתנגש באסטרואידים (כאשר תתנגש תופיע הודעה וייעלם האסטרואיד שהתנגשנו בו).

שלב ד

משימה 1 - הוספת טורפדואים

כדי לתקוף את האסטרואידים על החללית שלנו להיות בעלת יכולת לירות טורפדואים! לחיצה על המקש רווח מסמנת לנו שאנו רוצים לבצע יריה. 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).**

הערה 1: בשביל להוסיף את הטורפדו למשחק עליכם להשתמש בפונקציה register_torpedo שחתימתה היא:

```
def register_torpedo(torpedo)
```

הערה 2: הכיוון והמיקום ההתחלתי של הטורפדו צריך להיות הכיוון של החללית בזמן היריה.

הערה 3: שימו לב שגם כאן אתם תהיו אחראיים על הזזת הטורפדואים. בשביל לצייר טורפדו כלשהו על המסך ישנה הפונקציה draw_torpedo שנמצאת באובייקט Screen, חתימת הפונקציה הינה:

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

משימה 2 - ביצוע יריה

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

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

הערה 1: עבור ציר Y נשתמש ב-sin במקום cos.

הערה 2: המהירות הינה קבועה ואינה משתנה לכל אורך חייו של הטורפדו.

הערה 3: 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 האם טורפדו התנגש באסטרואיד, הרדיוס של טורפדו הינו קבוע וערכו 4 (הרדיוס הזה ניתן בשביל הנוסחאת התנגשות).

במקרה בו טורדפו התנגש עם אסטרואיד (כלומר get_intersection החזיר ערך True) האסטרואיד מתפוצץ אך לא נעלם! המשימות הבאות ידריכו אתכם בפעולות שצריכות להתקיים במקרה שבו ישנה התנגשות.

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

האסטרואיד מושמד ויקנה נקודות למשתמש 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 שזה אומר שעליכם לשמור את הניקוד של המשתמש, עדכון הנקודות על המסך מתבצע דרך המחלקה Screen דרך הפונקציה set_score אשר חתימתה הינה:

```
set_score(value)
```

<u>משימה 4.3 - פיצול האסטרואיד</u>

אם האסטרואיד הינו אסטרואיד גדול, כלומר גודלו גדול מ-1 (כלומר צריכה להיות לכם דרך לקבל את גודל האסטרואיד - <u>רמז</u> מתודה חדשה), האוסטרואיד ייתפצל לשניים בצורה הבאה:

שני האסטרואידים יתחילו עם אותן קואורדינטות כמו האסטרואיד הגדול יותר (לפני הפיצוץ) ובגודל קטן יותר (<u>חשוב!</u> אסטרואיד בגודל 3 יהפוך לשניים בגודל 2 ואסטרואיד בגודל 1 ייעלם מהמסך), מה שיישתנה בין האסטרואידים הינם המהירויות שלהם. חישוב המהירות החדשה, על כל ציר, תינתן לפי הנוסחה:

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

<u>הערה 1:</u> בכדי ליצור תנועה בכיוונים נגדיים תבחרו את אחד האסטרואידים ותכפלו את המהירות המקורית שלו (על שני הצירים) בערך 1-.

<u>הערה 2:</u> את הטורפדואים שפגעו באסטרואידים יש להסיר מהמסך.

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

<u>משימה 5 - זמן חיים לטורפדו</u>

כל טורפדו אחרי כמה זמן יוצא מתפעול, בשביל למדוד את זמן החיים של הטורפדו אתם צריכים למדוד את מס׳ הסבבים של המשחק שבו הוא חי, או במונח אחר 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 ה-id של הטורפדו המוסר.

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

יש להגביל את כמות הטורפדואים שיכולים להיות במשחק ל-15 בכל רגע נתון. כלומר, אם החללית ירתה 15 טורפדואים, היא לא תוכל לירות אף טורפדו חדש עד שאחד מהטורפדואים הקיימים "יצא מתפעול" (עבר את זמן החיים שלו).

<u>סיכום שלב ד' -</u> כרגע אם תריצו את המשחק צריכה להיות לכם חללית שעפה במסך עם אסטרואידים, החללית יכולה להסתובב ולהאיץ את מהירותה ובנוסף להתנגש באסטרואידים (כאשר תתנגש תופיע הודעה וייעלם האסטרואיד שהתנגשנו בו). בנוסף החללית יכולה לירות טורפדואים שיעזרו לה לנצח במשחק.

<u>שלב ה'</u>

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

הפסקת המשחק צריכה להתקיים באחד משלושת המקרים הבאים: (1) אם פוצצנו את כל האסטרואידים במשחק או (2) נגמרו לנו החיים או (3) לחצו על מקש היציאה 'q'.
בכל אחד מהמקרים צריכה להיות מודפסת הודעה המסמנת את סיבת היציאה.

<u>הערה 1:</u> כמו מקודם תוכן ההודעה לא משנה, העיקר הכוונה.

<u>הערה 2:</u> על מנת לסיים את המשחק ולסגור את הגרפיקה יש להשתמש בפונקציה end_game הנמצאת תחת המחלקה Screen (פונקציה זו לא מקבלת פרמטרים). לאחר קריאה לפונקציה זו תקראו גם ל-sys.exit.

<u>סיכום שלב ה' -</u> שימו לב שבשלב זה אמור להיות לכם משחק מוכן אתם מוזמנים לשחק בו להנאתכם :)

חלק ו' - כתיבת קובץ README

אתם מתבקשים לכתוב 3 שיקולים שהיו לכם בעיצוב המשחק, כלומר אתם מתבקשים לכתוב 3 החלטות שהייתם צריכים להחליט. לכל אחת מההחלטות האלו עליכם לרשום אלטרנטיבה אחרת ועבורה לציין מה הייתרון שלה על האפשרות שבחרתם ומדוע בחרתם שלא לממש אותה.

<u>הערות כלליות</u>

1. **שימו לב איפה אתם ממקמים את הקבועים של התוכנית, למשל האם מספר הטורפדואים שחללית יכולה לירות צריך להשמר במחלקה של חללית 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 לדחיה במועד ההגשה) יש לכתוב 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 ובודמה 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 ex9 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 ידרס!

- 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!