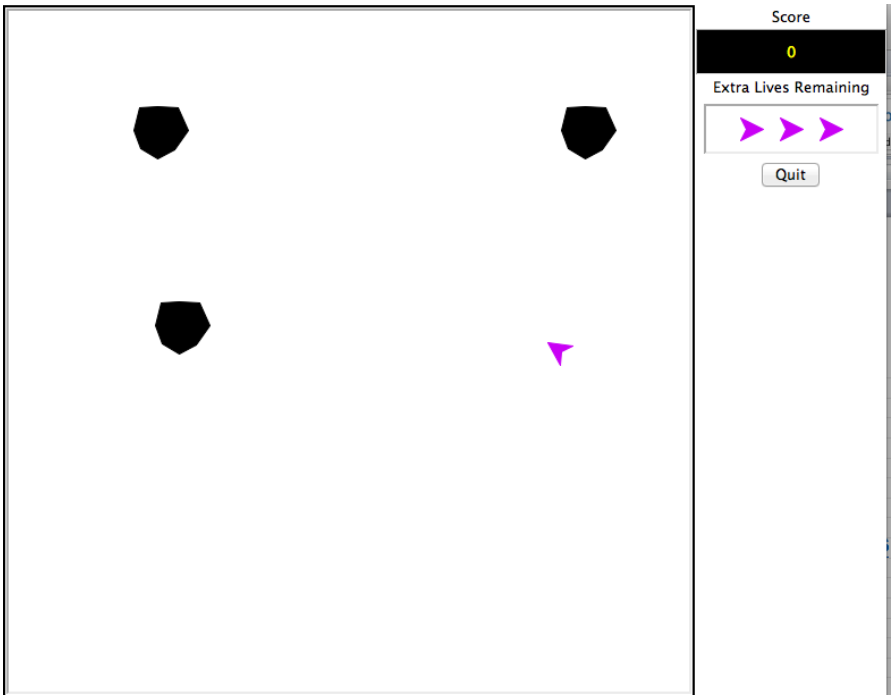
תרגיל 8 – משחקים עם אובייקטים

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:

- אתם מוזמנים להרחיב את ההתנהגות הסטנדרטית (בהמשך) כרצונכם, רק אנא התייחסו לכך בקובץ ה README.
- התרגיל אומנם נראה ארוך אך זאת רק מראית עין ☺

לצורך פתרון התרגיל מימשנו עבורכם מספר מחלקות בהם תצטרכו להשתמש. כל מחלקה מכילה פונקציות אשר ייעזרו לכם. המחלקות הינן:

1. BaseObject - מחלקה המייצגת כל אחד מהישויות במשחק (חללית, אסטרואיד, יריות).
2. GameMaster - מחלקה השומרת את מצב המשחק הנוכחי ואחראית על הגרפיקה.
3. OriginalGame - מכיל מתודות עבור הבונוסים.

<u>הערה:</u> המחלקה הרביעית הינה ShapesMaster בה אין לכם שימוש בכלל לכל אורך התרגיל.

בשביל לקרוא יותר על המחלקות ועל הפונקציות שהן חושפות אתם מוזמנים לפתוח את הקובץ index.html הנמצא בקובץ api.zip .

בשביל להתחיל לשחק במשחק הריצו את הפקודה:

```
python3 asteroidsMain.py
```

רקע מקדים:

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 asteroidsMain.py, את הפונקציה הזו **אתם** תצטרכו להשלים לפי המשימות 2-9 (בסדר הזה) הנתונות 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 היכרות שלכם עם עבודה עם אובייקטים, בתרגילים הבאים כבר תצטרכו לייצר אובייקטים ומחלקות חדשות משלכם.

משימות התרגיל

הערה 1: ביצוע המשימות לפי סדר יבטיח לכם פעולה נכונה של המשחק, אך ישנן הרבה דרכים שניתן לבצע זאת - המשימות למטה הינן הדרכה בלבד.

הערה 2: המטרה של התרגיל הינה שתתנסו בעבודה עם אובייקטים, 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.

הערה 3: את כל הפונקציות יש לממש בקובץ asteroidsMain.py.

הערה 4: כל תוספת שתבצעו בתרגיל צריכה להיות מצויינת בקובץ ה-README.

הערה 5: תקראו קודם את קובץ ה-API שמסופק לכם, כל הפונקציונאליות נמצאת בו ועל כן שאלות כמו "איך ניגשים לקואורדינטת ה-X של האובייקט" לא ייענו.

משימה 1 - תזוזה של האובייקטים

דבר ראשון שנרצה לבצע הינו לגרום לאובייקטים במשחק לזוז, לשם כך תממשו את הפונקציה moveObject. הפונקציה תקבל קלט אחד (לפחות) וזהו האובייקט שברצוננו להזיז.

בשביל להזיז את האובייקט O נצטרך לדעת מס' פרמטרים: (1) המהירות שלו על שני הצירים ; (2) המיקום שלו על הצירים ו-(3) גודל המסך. הנוסחה לחישוב המיקום החדש של האובייקט בכל ציר תהיה אם כך:

$$NewCord_{axis} = \left(speed_{axis} + OldCord_{axis} - AxisMinCord\right)\%D_{axis} + AxisMinCord$$

כאשר מתקיים ש= $D_{axis} = AxisMaxCord - AxisMinCord$.

עכשיו לאחר שממימשנו את הפונקציה הזו נוכל לקרוא לה עבור כל אסטרואיד, טורפדו והחללית שלנו.

הערה 1: שימו לב שהגדלים של המסך נשמרים עבורכם בפונקציית __init__ של האובייקט GameRunner.

הערה 2: שימו לב שלכל אובייקט ישנה מהירות על כיוון הציר X ועל כיוון הציר Y, גישה למהירות הזו תתבצע בעזרת המתודה getSpeedX או getSpeedY של האובייקט. בנוסף גישה לקואורדינטת ה-X או Y תתבצע באמצעות הקריאה למתודות getXCor ו-getYCor בהתאמה.

משימה 2 - הזזת האסטרואידים

בשביל לגשת לאובייקטים במשחק הגדרנו עבורכת בפונקצייה __init__ את האובייקט self.game. אובייקט זה מכיל את תמונת המצב של המשחק ומכיל פונקציה לקבלת כל האסטרואידים כתור רשימה.

משימה 3 - הזזת החללית

התזוזה של החללית צריכה להיות בהתאם למקשים שנלחצו ע"י המשתמש באופן הבא: (1) כשנלחץ ימינה נרצה להגדיל את הזווית של החללית ; (2) כשנלחץ שמאלה נקטין את הזווית ו-(3) כאשר נלחץ על למעלה נרצה להאיץ.

התאוצה תגרום לשינוי במהירות של החללית על כל אחד מהצירים, עבור ציר ה-X היא תינתן לפי הנוסחה הבאה:

$NewSpeed_{axis} = CurrentSpeed_{axis} + \cos\left(ShipAngleRadians\right)$, עבור ציר Y נשתמש ב-sin במקום cos.

אחרי שעידכנו את מצב החללית (אין צורך לתמוך בלחיצה של יותר ממקש תזזוזה אחד בו זמנית) נזיז אותה באמצעות הפונקציה שמימשנו במשימה 1.

הערה 1: שימו לב שקיימת פונקציה לבדיקת לחיצה על כל מקש תחת self.game.

הערה 2: בשביל לגשת לחללית המשחקת במשחק ניתן לקרוא לפונקציה getShip הקיימת בתוך game.self.

הערה 3: המרה של הזווית ממעלות לרדיאנים מושארת כמשימה לכם.

משימה 4 - ביצוע יריה של טורפדו

כדי לתקוף את האסטרואידים על החללית שלנו להיות בעלת יכולת לירות טורפדואים! אך מכיוון שהמלאי שלה מוגבל היא יכולה לירות לא יותר מ-20 טורפדואים בו זמנית. המהירות של הטורפדו עבור ציר ה-X תינתן לפי הנוסחה הבאה:

$NewSpeed_{axis} = CurrentSpeed_{axis} + 2 \cdot \cos(ShipAngleRadians)$, עבור ציר Y נשתמש ב-sin במקום cos.

משימה 5 - הזזת הטורפדואים

קיימת באובייקט self.game פונקציה לקבלת רשימת כל הטורדפדואים שקיימים במשחק, הפעולה שצריך לבצע הינה זהה לפעולה של הזזת האסטרואידים.

הערה: שימו לב שלטורפדו יש זמן חיים, הניתן ע"י הפונקציה getLifeSpan, טורפדו צריך להיות **מסומן** להסרה עם אורך החיים שלו הינו קטן או שווה מ=0.

משימה 6 - פיצוץ אסטרואידים

כאשר טורפדו ששלחנו מתנגש עם אסטרואיד (לבדיקת התנגשות תסכלו על ה-API של GameMaster) האסטרואיד מושמד ויקנה נקודות למשתמש באופן הבא:

אסטרואיד בגודל 3 = 20 נקודות

אסטרואיד בגודל 2 = 50 נקודות

אסטרואיד בגודל 1 = 100 נקודות

אם האסטרואיד הינו אסטרואיד גדול, כלומר גודלו גדול מ-1, האוסטרואיד ייתפצל לשניים בצורה הבאה:

שני האסטרואידים יתחילו עם אותן קואורדינטות כמו האסטרואיד הגדול יותר (לפני הפיצוץ), מה שיישתנה בין האסטרואידים הינם המהירויות שלהם. חישוב המהירות החדשה, על כל ציר, תינתן לפי הנוסחה:

$$NewSpeed_{axis} = \frac{TorpedoSpeed_{axis} + CurrentSpeed_{axis}}{\sqrt{Speed^2{}_x + Speed^2{}_y}} .$$

<u>הערה 1:</u> בכדי ליצור תנועה בכיוונים נגדיים תבחרו את אחד האסטרואידים ותשנו לו את המהירות לשלילית בשני הצירים.

<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).

משימה 7 - הסרה של טורפדואים ואסטרואידים

לאחר שטורפדו סיים את חייו (או פגע באסטרואיד) נצטרך להסיר אותו מהמסך, לשם כך יש לכם פונקציה מתאימה תחת האובייקט self.game: הפונקציה מקבלת כקלט רשימה של טורפדואים אשר צריך להסיר.

הסרה של אסטרואידים מהמסך (כאלו שהתפוצצו) תתבצע בעזרת פונקציה דומה תחת self.game.

משימה 8 - פגיעה של החללית באסטרואיד

אם החללית תפגע באסטרואיד אז צריכה להופיע הודעה (שוב הסברים תחת self.game) אשר אומרת שאיבדנו חיים. צריכה להופיע הודעה אחת בלבד, גם אם פגענו ביותר מאסטרואיד 1 בו זמנית.

הערה 1: נוסח ההודעה לא משנה, הוא יכול להיות כמו בתרגיל או כל הודעה קונסטרוקטיבית אחרת.

הערה 2: האסטרואיד שפגענו בו צריך להיות מוסר לגמרי ולא להתפוצץ לשני חלקים כמו שמתרחש בפגיעה של טורפדו.

משימה 9 – הפסקת/סיום המשחק

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

הערה 1: כמו מקודם תוכן ההודעה לא משנה, העיקר הכוונה.

הערה לסיכום

הפונקציה game_loop שמימשתם מהווה איטרציה אחת של לולאת המשחק, אתם לא צריכים לדאוג לקרוא לפונקציה הזו שוב.

בונוסים

בונוס 1: (מקסימום 10 נקודות)

אתם יכולים לממש סוגים שונים של אובייקטים נוספים למשחק, ניתן לקרוא על כך ב-API של OriginalGame, כאשר את האובייקט הנ"ל ניתן להשיג בעזרת מתודה שב-self.game.

לקבלת הבונוס יש לממש סוג חדש של טורפדואים (למשל טורפדו שזז בתנועה מעגלית), בשביל להוסיף טורפדו מסוג חדש שייצרתם אתם צריכים להשתמש בפונקציה המתאימה. הטורפדו החדש לאחר שיוסף ייתנהג כאילו הוא טורפדו רגיל (מבחינת חוקי המשחק של טורפדו) על כן חשבו על כך בהתאם.

בונוס 2: (מקסימום 5 נקודות)

הוסיפו אפשרות לשליחת מספר האסטרואידים ההתחלתי שיהיה במשחק.

קובץ ה-README

1. קובץ תארו האם מימשתם את הבונוסים בתרגיל, אם כן אז הסבירו איך מימשתם כל אחד מהם.
2. שימו לב שה-Usage בתרגיל הנ"ל זה בדיוק השורה שבה אתם מפעילים את המשחק, כך שאם מימשתם את בונוס 2 אז ה-Usage צריך להשתנות בהתאם.
3. תארו בקובץ איך הייתה חווית השימוש ב-API שניתן לכם.

נהלי הגשה

הלינק להגשה של התרגיל הינו תחת ex8.

בתרגיל זה עליכם להגיש את הקבצים הבאים:

1. asteroidsMain.py – עם המימושים שלכם לפונקציית game_loop.
2. README (על פי פורמט ה-README לדוגמא שיש באתר הקורס, ועל פי ההנחיות לכתיבת README המפורטות בקובץ נהלי הקורס).

יש ליצור קובץ tar הנקרא ex8.tar המכיל בדיוק את שני הקבצים הנ"ל (בנוסף לקבצים הנוספים שייצרתם – אם הם קיימים), בעזרת פקודת ה-shell הבאה:

tar cvf ex8.tar asteroidsMain.py README, אם קיימים עוד קבצים שייצרתם הוסיפו את השמות שלהם אחרי קובץ ה-README.

הערה: מומלץ לבדוק את קובץ ה-tar שיצרתם על ידי העתקת התוכן שלו לתיקייה נפרדת ופתיחתו (extract) בעזרת ביצוע הפקודה: tar xvf ex8.tar, 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 7 הריצו את השורה הבאה ב-shell:

~/intro2cs/bin/presubmit/ex8 ex8.tar

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, ומעבר של בדיקות הסקריפט לא מבטיח את תקינותה של התוכנית! עליכם לוודא בעצמכם שהתוכנית שלכם פועלת כפי שדרוש.

הגשת קובץ tar

עליכם להגיש את הקובץ ex8.tar בקישור ההגשה של תרגיל 8! לאחר הגשת התרגיל, ניתן ומומלץ להוריד אותו ולוודא כי הקבצים המוגשים הם אלו שהתכוונתם להגיש וכי הקוד עובד על פי ציפיותיכם.

לאחר שאתם מגישים את התרגיל באתר הקורס, תוך מספר שניות יווצר הקובץ submission.pdf . עליכם לבדוק שהכל תקין בקובץ submission.pdf! ואם יש בעיה כלשהי בקבצים שלכם שבאה לידי ביטוי בקובץ ה-pdf עליכם לתקן אותה, גם אם לא נאמר שום דבר בפירוש בתרגיל לגבי זה, כך שקובץ ה-pdf שמיוצר מהתרגיל שלכם יהיה תקין לגמרי. וודאו שאין שורות ארוכות מדי בקוד שלכם שנחתכות בקובץ ה-pdf, ושהקובץ מסודר ורואים בו את הכל בצורה טובה וברורה. זכרו את ההגבלה של 79 תווים לכל היותר בשורה (כולל הערות).

בהצלחה ☺!