

# Contents

1	Basic Test Results	2
2	README	3
3	AUTHORS	4
4	asteroid.py	5
5	asteroids main.py	7
6	screen.py	12
7	ship.py	20
8	torpedo.py	22

# 1 Basic Test Results

```
1 Starting tests...
2 Thu Dec 24 00:07:40 IST 2015
3 e87f4ea2769cbf2cfa4d7ac0d88262d827245449 -
4
5
6 Archive: /tmp/bodek.m4tG4o/intro2cs/ex9/elinorperl/presubmission/submission
7   inflating: src/asteroid.py
8   inflating: src/asteroids_main.py
9   extracting: src/AUTHORS
10  inflating: src/screen.py
11  inflating: src/ship.py
12  inflating: src/torpedo.py
13  inflating: src/README
14
15 Testing README...
16 Done testing README...
17
18 Listing AUTHORS...
19 elinorperl
20 ynatovich
21
22
23 Tests completed
24
25 Additional notes:
26
27 No presubmission tests of the code this week.
```

## 2 README

```
1  elinorperl
2  329577464
3  Elinor Perl
4
5  ynatovich
6  204858914
7  Yifat Natovitch
8
9
10 I discussed the exercise with Talya Adams, Yonaton Iluz
11
12 =====
13 =  README for ex9: Asteroids OOP  =
14 =====
15
16
17 =====
18 =  Description:  =
19 =====
20
21 In this exercise, using OOP, we defined three different classes "Ship",
22 "Asteroid, and "Torpedo", each playing a part in the game we created -
23 Asteroids. Each class was defined according to the attributes we expected
24 from them in the game in which the object is for the ship to avoid and destroy
25 the asteroids, defeating them by using the torpedos which split the asteroids
26 into smaller asteroids until they are small enough to dissappear.
27
28 =====
29 =  Special Comments  =
30 =====
31
32 We used stackoverflow.com
33
34 Throughout programming this game, we encountered obstacles about how to
35 proceed with our game, amongst them:
36
37 1. When creating a function to move our ship, we were unsure in which
38 class it belonged. On the one hand, we're moving the ship and movement
39 is an attribute belonging to itself. On the other, in order to display
40 our movement on the screen we needed the "intermediary" game which
41 acts as the bridge between the two classes. In the end, we decided
42 upon placing the ship's movement in its own class and calling it
43 to our intermmmediary when necessary.
44
45 2. In the class "Torpedo", we debated on whether to define a constant in the
46 initial creation of itself, or to give the ability to an outside user to change
47 it throughout the game. The advantages of the changability throughout give the
48 user the flexibility to use the constant. We decided against this method, because
49 we wanted to save the initial attributes according the game rules.
50
51 3. When ending the game, we didn't know whether to make a one function
52 to end the game or to split it into two, to give it a "cleaner" look,
53 rather than repeating the same actions three times. The layout we ended
54 with is longer and maybe not necessary but we felt it was more sleek.
```

## 3 AUTHORS

1 elinorperl, ynatovich

## 4 asteroid.py

```
1  from random import randrange, uniform
2  from screen import Screen as Scr
3  import copy
4  import math
5  from ship import Ship
6
7  class Asteroid:
8
9      DELTA_X = Scr.SCREEN_MAX_X - Scr.SCREEN_MIN_X
10     DELTA_Y = Scr.SCREEN_MAX_Y - Scr.SCREEN_MIN_Y
11     RADIUS_COEFFIC = 10
12     NORM_FACTOR = -5
13     BIG = 3
14     MIDDLE = 2
15     SMALL = 1
16
17     def __init__(self):
18         self.__x_speed = uniform(-5, 5)
19         self.__y_speed = uniform(-5, 5)
20         self.__x_position = randrange(Scr.SCREEN_MIN_X, Scr.SCREEN_MAX_X)
21         self.__y_position = randrange(Scr.SCREEN_MIN_Y, Scr.SCREEN_MAX_Y)
22         self.__size = 3
23         self.__radius = self.__size * self.RADIUS_COEFFIC + self.NORM_FACTOR
24
25     def get_x_position(self):
26         return copy.copy(self.__x_position)
27
28     def get_y_position(self):
29         return copy.copy(self.__y_position)
30
31     def get_x_speed(self):
32         return copy.copy(self.__x_speed)
33
34     def get_y_speed(self):
35         return copy.copy(self.__y_speed)
36
37     def get_size(self):
38         return copy.copy(self.__size)
39
40     def change_position(self, new_position):
41         """
42         Changes the position of the asteroid according to the new position
43         input.
44         """
45         self.__x_position, self.__y_position = new_position[0], new_position[1]
46
47     def change_size(self, size):
48         """
49         Changes the size of the asteroid according to the size input.
50         """
51         self.__size = size
52
53
54     def move(self):
55         """
56         This function calculates our new coordinate of the position for our
57         asteroid.
58         """
59         self.__x_position = ((self.get_x_speed() + self.get_x_position() -
```

```

60         Scr.SCREEN_MIN_X) % self.DELTA_X + Scr.SCREEN_MIN_X)
61     self.__y_position = ((self.get_y_speed() + self.get_y_position() -
62         Scr.SCREEN_MIN_Y) % self.DELTA_Y + Scr.SCREEN_MIN_Y)
63
64     def get_interaction(self, obj):
65         distance_x = math.pow((obj.get_x_position() - self.get_x_position()),2)
66         distance_y = math.pow((obj.get_y_position() - self.get_y_position()),2)
67         distance = math.sqrt(distance_x + distance_y)
68         if distance <= self.__radius + obj.RADIUS:
69             return True
70         return False
71
72     def speed_after_interaction(self, torpedo, direction):
73         """
74         A function that calculates the speed of the new asteroids that were
75         created after a collision with a torpedo, where one of the parameters
76         is direction, therefore allowing us to later differentiate between
77         the two directions of the new asteroids. The asteroid splits into two,
78         allowing us to use the indexes for 0 and 1 as our direction guide. 0
79         for the positive direction and 1 for the negative.
80         """
81         speed_denominator = math.sqrt(math.pow(self.__x_speed,2) + \
82             math.pow(self.__y_speed,2))
83         self.__x_speed = ((torpedo.get_x_speed() + self.get_x_speed()) /
84             speed_denominator)
85         self.__y_speed = ((torpedo.get_y_speed() + self.get_y_speed()) /
86             speed_denominator)
87         if direction == 1:
88             self.__x_speed *= -1
89             self.__y_speed *= -1

```

## 5 asteroids main.py

```
1  from screen import Screen
2  import sys
3  from ship import Ship
4  from asteroid import Asteroid
5  from torpedo import Torpedo
6  import copy
7  from random import randrange
8
9  DEFAULT_ASTEROIDS_NUM = 5
10
11
12  class GameRunner:
13
14      DELTA_X = Screen.SCREEN_MAX_X - Screen.SCREEN_MIN_X
15      DELTA_Y = Screen.SCREEN_MAX_Y - Screen.SCREEN_MIN_Y
16      MAX_TORPEDOS = 15
17
18      def __init__(self, asteroids_amnt=DEFAULT_ASTEROIDS_NUM):
19          self._screen = Screen()
20          self.screen_max_x = Screen.SCREEN_MAX_X
21          self.screen_max_y = Screen.SCREEN_MAX_Y
22          self.screen_min_x = Screen.SCREEN_MIN_X
23          self.screen_min_y = Screen.SCREEN_MIN_Y
24          self._ship = Ship()
25          self._asteroids_amnt = asteroids_amnt
26          self._asteroids = []
27          self._torpedos = []
28          self._score = 0
29          self._screen.set_score(self._score)
30          self.create_asteroids()
31
32      def get_score(self):
33          return copy.copy(self._score)
34
35      def create_asteroids(self):
36          """
37          In this function, we created a list of the asteroids according to the
38          amount give (if none was given, according to the default amount) and
39          registered each one.
40          """
41          for i in range(self._asteroids_amnt):
42              asteroid = Asteroid()
43              self.get_random_position_asteroid(asteroid)
44              self._asteroids.append(asteroid)
45              self._screen.register_asteroid(asteroid, asteroid.get_size())
46
47      def get_random_position_asteroid(self, asteroid):
48          """
49          This function places the asteroid in a random location, and if the
50          asteroids location lands on the spaceship's initial location, the
51          function recursively relocates the asteroid.
52          """
53          asteroid.change_position((randrange(Screen.SCREEN_MIN_X,
54                                              Screen.SCREEN_MAX_X),\
55                                  randrange(Screen.SCREEN_MIN_X,
56                                              Screen.SCREEN_MAX_X)))
57          while ((asteroid.get_x_position(), asteroid.get_y_position()) == \
58                (self._ship.get_x_position(), self._ship.get_y_position())):
59              self.get_random_position_asteroid(asteroid)
```

```

60
61 def run(self):
62     self._do_loop()
63     self._screen.start_screen()
64
65 def _do_loop(self):
66     # You don't need to change this method!
67     self._game_loop()
68
69     # Set the timer to go off again
70     self._screen.update()
71     self._screen.ontimer(self._do_loop, 5)
72
73 def operate_torpedo(self):
74     """
75     This function operates the torpedo's functionality. Calling the class
76     Torpedo and inputting ship's attributes (serving as the ship and
77     torpedo's bridge of connection), adding each torpedo to the list
78     created and drawing it on the screen.
79     """
80     torpedo = Torpedo(self._ship.get_x_position(),
81                       self._ship.get_y_position(),
82                       self._ship.get_x_speed(),
83                       self._ship.get_y_speed(),
84                       self._ship.get_heading())
85     self._torpedos.append(torpedo)
86     self._screen.register_torpedo(torpedo)
87     self._screen.draw_torpedo(torpedo, torpedo.get_x_position(),
88                              torpedo.get_y_position(),
89                              torpedo.get_direction())
90
91 def game_movements(self):
92     """
93     This function checks if the user pressed left, right or up and
94     depending on the input, it returns the function accordingly.
95     """
96     if Screen.is_left_pressed(self._screen):
97         self._ship.change_heading_left()
98     elif Screen.is_right_pressed(self._screen):
99         self._ship.change_heading_right()
100    elif Screen.is_up_pressed(self._screen):
101        self._ship.accelerated_ship()
102    elif Screen.is_space_pressed(self._screen):
103        if len(self._torpedos) < self.MAX_TORPEDOS:
104            self.operate_torpedo()
105
106 def draw_objects(self):
107     """
108     This function calls the functions draw ship and draw (each) asteroid
109     on the screen.
110     """
111     self._screen.draw_ship(self._ship.get_x_position(),
112                           self._ship.get_y_position(),
113                           self._ship.get_heading())
114     for asteroid in self._asteroids:
115         self._screen.draw_asteroid(asteroid, asteroid.get_x_position(),\
116                                   asteroid.get_y_position())
117     for torpedo in self._torpedos:
118         self._screen.draw_torpedo(torpedo, torpedo.get_x_position(),
119                                   torpedo.get_y_position(),
120                                   torpedo.get_direction())
121
122 def move_asteroid(self):
123     """
124     This function moves each asteroid from the list of asteroids that we
125     created, moving each asteroid in turn and having them appear on the
126     screen.
127     """

```



```

128         for asteroid in self._asteroids:
129             asteroid.move()
130
131     def move_torpedos(self):
132         """
133         This function moves each asteroid from the list of asteroids that we
134         created, moving each asteroid in turn and having them appear on the
135         screen.
136         """
137         for torpedo in self._torpedos:
138             torpedo.move()
139
140     def interaction_ship(self, asteroid):
141         """
142         This function deals with the interaction between the ship and
143         asteroids by removing one of the users lives, and displaying a message
144         to the user to notify them that they hit an asteroid and thereafter
145         removing them.
146         """
147         Screen.remove_life(self._screen)
148         self._ship.update_life()
149         # in the event that there are two collision in one turn:
150         if self._ship.get_life() == 0:
151             self.ending("Out of lives", "Maybe next time...")
152         self._screen.show_message("BOOM!", "You hit"
153                                  " an asteroid. BE CAREFUL!")
154         self._asteroids.remove(asteroid)
155         self._screen.unregister_asteroid(asteroid)
156
157     def interaction_torpedo(self, asteroid, torpedo):
158         """
159         This function deals with the occurrence of interaction with torpedos
160         and asteroids, removing the torpedo from the screen, and updating
161         the user's score.
162         """
163         self._torpedos.remove(torpedo)
164         self._screen.unregister_torpedo(torpedo)
165         self.update_score(asteroid)
166         self.split_asteroid(asteroid, torpedo)
167
168     def create_new_asteroids(self, asteroid, torpedo, size):
169         """
170         This function splits the current asteroid to 2 new smaller ones and
171         defines its attributes and position on the screen.
172         """
173         get_x = asteroid.get_x_position()
174         get_y = asteroid.get_y_position()
175         for i in range(2):
176             asteroid_i = Asteroid()
177             asteroid_i.change_position((get_x, get_y))
178             asteroid_i.change_size(size)
179             asteroid_i.speed_after_interaction(torpedo, i)
180             self._asteroids.append(asteroid_i)
181             self._screen.register_asteroid(asteroid_i, asteroid_i.get_size())
182
183     def split_asteroid(self, asteroid, torpedo):
184         """
185         This function splits the asteroid when it gets hit. and deletes the
186         asteroid when it has reached its smallest size and has been hit as
187         well.
188         """
189         self._asteroids.remove(asteroid)
190         self._screen.unregister_asteroid(asteroid)
191
192         if asteroid.get_size() == asteroid.BIG:
193             self.create_new_asteroids(asteroid, torpedo, asteroid.MIDDLE)
194         if asteroid.get_size() == asteroid.MIDDLE:
195             self.create_new_asteroids(asteroid, torpedo, asteroid.SMALL)

```

```

196
197 def update_score(self, asteroid):
198     """
199     A function that updates the score according to the asteroid's size.
200     """
201     if asteroid.get_size() == asteroid.BIG:
202         self._score += 20
203     if asteroid.get_size() == asteroid.MIDDLE:
204         self._score += 50
205     if asteroid.get_size() == asteroid.SMALL:
206         self._score += 100
207     self._screen.set_score(self._score)
208
209 def interaction(self):
210     """
211     Using a copied version of our asteroid_list, we checked if there was a
212     collision with each asteroid and the spaceship. If there was a inter-
213     action, the spaceship removes a life, and displays a message that
214     there has been a collision and removes an asteroid from our asteroid
215     list.
216     """
217     copied_asteroid = copy.copy(self._asteroids)
218     copied_torpedos = copy.copy(self._torpedos)
219     for asteroid in copied_asteroid:
220         if asteroid.get_interaction(self._ship):
221             self.interaction_ship(asteroid)
222     for asteroid in copied_asteroid:
223         for torpedo in copied_torpedos:
224             if asteroid.get_interaction(torpedo):
225                 # if there is a collision, loop continues to other objects
226                 if (asteroid in self._asteroids) and \
227                     (torpedo in self._torpedos):
228                     self.interaction_torpedo(asteroid, torpedo)
229
230 def update_lifespan_torpedos(self):
231     """
232     Updates the lifespan of each torpedo on our list
233     """
234     copied_torpedos = copy.copy(self._torpedos)
235     for torpedo in copied_torpedos:
236         torpedo.update_lifespan()
237         if torpedo.get_lifespan() == torpedo.LIFESPAN_MAX:
238             self._torpedos.remove(torpedo)
239             self._screen.unregister_torpedo(torpedo)
240
241 def end_game(self):
242     """
243     Returns message according to the reason for finishing the game and
244     calls to the function that displays the message and closes the tab.
245     """
246     if len(self._asteroids) == 0:
247         self.ending("WINNER!!!", "YOU ARE THE CHAMPION MY FRIEND! YOU "
248                     "HAVE DESTROYED ALL THE ASTEROIDS :)")
249     if self._ship.get_life() == 0:
250         self.ending("Out of lives", "Maybe next time...")
251     if self._screen.should_end():
252         self.ending("QUITTER", "Byebye!")
253
254 def ending(self, title, msg):
255     """
256     This function ends the game by showing the message with the reason
257     for ending, then closes the screen.
258     """
259     self._screen.show_message(title, msg)
260     self._screen.end_game()
261     sys.exit()
262
263 def _game_loop(self):

```

```

264         '''
265         This is the function that ties the whole game together, calling each
266         function in turn, and creating each round of the game.
267         '''
268         self._ship.move()
269         self.move_asteroid()
270         self.move_torpedos()
271         self.game_movements()
272         self.draw_objects()
273         self.interaction()
274         self.update_lifespan_torpedos()
275         self.end_game()
276
277
278     def main(amnt):
279         runner = GameRunner(amnt)
280         runner.run()
281
282     if __name__ == "__main__":
283         if len(sys.argv) > 1:
284             main( int( sys.argv[1] ) )
285         else:
286             main(DEFAULT_ASTEROIDS_NUM)

```

## 6 screen.py

```
1 import sys
2 import tkinter
3 import tkinter.messagebox
4
5 from turtle import *
6
7 class ShapesMaster:
8     ASTEROID_BASE_SHAPE = "asteroid%d"
9     SHIP_SHAPE = "ship"
10    TORPEDO_SHAPE = "torpedo"
11
12    ASTEROIDS_TYPES = 3
13
14    ASTEROID_3_LAYOUT = ((-20, -16), (-21, 0), (-20, 18), (0, 27), (17, 15),
15                        (25, 0), (16, -15), (0, -21))
16
17    ASTEROID_2_LAYOUT = ((-15, -10), (-16, 0), (-13, 12), (0, 19), (12, 10),
18                        (20, 0), (12, -10), (0, -13))
19
20    ASTEROID_1_LAYOUT = ((-10, -5), (-12, 0), (-8, 8), (0, 13), (8, 6), (14, 0), (12, 0), (8, -6), (0, -7))
21
22    ASTEROIDS_LAYOUTS = [ASTEROID_1_LAYOUT, ASTEROID_2_LAYOUT, ASTEROID_3_LAYOUT]
23
24    SHIP_LAYOUT = ((-10, -10), (0, -5), (10, -10), (0, 10))
25
26    TORPEDO_LAYOUT = ((-2, -4), (-2, 4), (2, 4), (2, -4))
27
28    def __init__(self, screen):
29        """
30        This initializes the shapes controller, the screen passed is the screen
31        controlling the game, you should not call this method anywhere in your
32        code.
33        """
34        self.screen = screen
35        self._shapes = {}
36        self._updated = False
37        self._add_base_shapes()
38
39    def add_shape(self, name, cords, override = False):
40        if override or name not in self._shapes:
41            self._shapes[name] = cords
42            self.screen.register_shape(name, cords)
43
44    def _add_base_shapes(self):
45        for i in range(ShapesMaster.ASTEROIDS_TYPES):
46            self.add_shape(ShapesMaster.ASTEROID_BASE_SHAPE%(i+1), \
47                          ShapesMaster.ASTEROIDS_LAYOUTS[i])
48
49        self.add_shape(ShapesMaster.SHIP_SHAPE, ShapesMaster.SHIP_LAYOUT)
50        self.add_shape(ShapesMaster.TORPEDO_SHAPE, ShapesMaster.TORPEDO_LAYOUT)
51
52    def get_shapes_dict(self):
53        """
54        Returns a dictionary of all the shapes in the game in the format of
55        (name, coordinates).
56        You have no reason of calling this method anywhere in your code...
57        """
58        return self._shapes
59
```

```

60
61 class Screen:
62
63     SCREEN_MIN_X = -500
64     SCREEN_MIN_Y = -500
65     SCREEN_MAX_X = 500
66     SCREEN_MAX_Y = 500
67
68     def __init__(self):
69         """
70         This inits our graphics class.
71         """
72
73         self._boundKeys = []
74         self._init_keys_values()
75         self._init_graphics()
76         self._bind_keys()
77         self._screen.listen()
78
79         self._ship = self._get_ship_obj(self._cv)
80
81     def _init_keys_values(self):
82         self._specialTorpedFired = 0
83         self._rightClicks = 0
84         self._leftClicks = 0
85         self._upClicks = 0
86         self._fireClicks = 0
87         self._endGame = False
88         self._lives = []
89         self._asteroids = {}
90         self._torpedos = {}
91
92     def _init_graphics(self):
93         self._root = tkinter.Tk()
94         self._root.title("Asteroids!")
95         self._cv = ScrolledCanvas(self._root,600,600,600,600)
96         self._cv.pack(side = tkinter.LEFT)
97         self._t = RawTurtle(self._cv)
98
99         self._screen = self._t.getscreen()
100         self._screen.setworldcoordinates(
101             Screen.SCREEN_MIN_X,
102             Screen.SCREEN_MIN_Y,
103             Screen.SCREEN_MAX_X,
104             Screen.SCREEN_MAX_X
105         )
106         self._shapeMaster = ShapesMaster(self._screen)
107         shapes = self._shapeMaster.get_shapes_dict()
108
109         frame = tkinter.Frame(self._root)
110         frame.pack(side = tkinter.RIGHT,fill=tkinter.BOTH)
111
112         # add scores frame
113         self._score_val = tkinter.StringVar()
114         self._score_val.set("0")
115         scoreTitle = tkinter.Label(frame,text="Score")
116         scoreTitle.pack()
117         scoreFrame = tkinter.Frame(frame,height=2, bd=1, \
118             relief=tkinter.SUNKEN)
119         scoreFrame.pack()
120         score = tkinter.Label(scoreFrame,height=2,width=20,\
121             textvariable=self._score_val,fg="Yellow",bg="black")
122
123         #####
124
125         score.pack()
126
127         # Add Lives Frame

```

```

128     # livesTitle = tkinter.Label(frame, \
129     #     text="Extra Lives Remaining")
130     # livesTitle.pack()
131
132     # livesFrame = tkinter.Frame(frame, \
133     #     height=30,width=60,relief=tkinter.SUNKEN)
134     # livesFrame.pack()
135     # self._lives_canvas = ScrolledCanvas(livesFrame,150,40,150,40)
136     # self._lives_canvas.pack()
137     # livesTurtle = RawTurtle(self._lives_canvas)
138     # livesTurtle.ht()
139     # livesScreen = livesTurtle.getscreen()
140     # livesScreen.register_shape(ShapesMaster.SHIP_SHAPE, shapes[ShapesMaster.SHIP_SHAPE])
141
142     # Add Lives Frame
143     livesTitle = tkinter.Label(frame, \
144     #     text="Extra Lives Remaining")
145     livesTitle.pack()
146
147     livesFrame = tkinter.Frame(frame, \
148     #     height=30,width=60,relief=tkinter.SUNKEN)
149     livesFrame.pack()
150     livesCanvas = ScrolledCanvas(livesFrame,150,40,150,40)
151     livesCanvas.pack()
152     livesTurtle = RawTurtle(livesCanvas)
153     livesTurtle.ht()
154     livesScreen = livesTurtle.getscreen()
155     livesScreen.register_shape(ShapesMaster.SHIP_SHAPE, shapes[ShapesMaster.SHIP_SHAPE])
156
157     life1 = self._get_ship_obj(livesCanvas) # SpaceShip(livesCanvas,-35,0,0,0)
158     life2 = self._get_ship_obj(livesCanvas) #SpaceShip(livesCanvas,0,0,0,0)
159     life3 = self._get_ship_obj(livesCanvas) #SpaceShip(livesCanvas,35,0,0,0)
160
161     self._draw_object(life1,-35,0)
162     self._draw_object(life2,0,0)
163     self._draw_object(life3,35,0)
164
165     self._lives = [life1, life2, life3]
166
167     self._t.ht()
168
169     quitButton = tkinter.Button(frame, text = "Quit", command=self._handle_exit)
170     quitButton.pack()
171
172     self._screen.tracer(0)
173
174     def ontimer(self, func, milli):
175         """
176         This method is used to create a repeating action in your game.
177
178         .. warning::
179
180             **You don't need to call this method, it was already called for you at the end of the main game loop.**
181
182         :param func: The function to repeat after **milli** milliseconds have passed
183         :type func: function
184         :param milli: The amount of milliseconds to wait before starting the given
185                     function
186         :type milli: int
187         """
188         self._screen.ontimer(func,milli)
189
190     def _bind_key(self, key, func):
191         """
192         This method is to allow you to add some functionality of your own,
193         it allows you to bind the provided function to the desired input key.
194
195         If there is already a function bound to this key it will do nothing.

```

```

196
197         :param key: A key to bind.
198         :type key: str
199         :param func: The function to bind
200         :type func: function
201         """
202
203         if key not in self._boundKeys:
204             self._screen.onkeypress(func, key)
205             self._boundKeys.append(key)
206
207     def _bind_keys(self):
208         self._bind_key("Left", self._handle_left)
209         self._bind_key("Right", self._handle_right)
210         self._bind_key("Up", self._handle_up)
211         self._bind_key("space", self._handle_space)
212         self._bind_key("q", self._handle_exit)
213         self._bind_key("s", self._handle_special_torpedo)
214
215     def _handle_special_torpedo(self):
216         self._specialTorpedFired += 1
217
218     def _handle_exit(self):
219         self._endGame = True
220
221     def _handle_left(self):
222         self._leftClicks += 1
223
224     def _handle_right(self):
225         self._rightClicks += 1
226
227     def _handle_up(self):
228         self._upClicks += 1
229
230     def _handle_space(self):
231         self._fireClicks += 1
232
233     def start_screen(self):
234         """
235         This is called to start our game (grphaics-wise).
236
237         .. warning::
238
239             **This method should not be called by you**
240         """
241         tkinter.mainloop()
242
243     def update(self):
244         """
245         This is called to update our game (grphaics-wise).
246
247         .. warning::
248
249             **This method should not be called by you**
250         """
251         self._screen.update()
252
253     def set_score(self, val):
254         """
255         Sets the current game score
256
257         :param val: The game score
258         :type val: int
259         """
260         self._score_val.set(str(val))
261
262     def _get_ship_obj(self, canvas):
263         ship = RawTurtle(canvas)

```

```

264         ship.shape(ShapesMaster.SHIP_SHAPE)
265         ship.color("purple")
266         return ship
267
268     def _get_asteroid_object(self, size):
269         asteroid = RawTurtle(self._cv)
270         asteroid.shape(ShapesMaster.ASTEROID_BASE_SHAPE%size)
271         return asteroid
272
273     def _get_torpedo_object(self):
274         torpedo = RawTurtle(self._cv)
275         torpedo.shape(ShapesMaster.TORPEDO_SHAPE)
276         torpedo.color("blue")
277         return torpedo
278
279     def _draw_object(self, obj, x, y, heading=None):
280         obj.penup()
281         obj.goto(x, y)
282         if heading:
283             obj.setheading(heading)
284         obj.pendown()
285
286     def remove_life(self):
287         """
288         Remove one icon of life (starts with 3 lives)
289         """
290         deadship = self._lives.pop()
291         deadship.ht()
292
293     def register_asteroid(self, asteroid, size):
294         """
295         This is called to register a new asteroid in our system
296
297         :param asteroid: This is your asteroid object
298         :type asteroid: Asteroid
299
300         :param size: The size of the asteroid (this should be in [1,2,3])
301         :type size: int
302         """
303         if size not in [1,2,3]:
304             print("Error: Wrong asteroid size: %d"%size)
305             sys.exit(0)
306         elif id(asteroid) in self._asteroids:
307             print("Error: Asteroid id (%d) already exists"%asteroid.id)
308             sys.exit(0)
309         asteroid_obj = self._get_asteroid_object(size)
310         self._asteroids[ id(asteroid) ] = asteroid_obj
311
312
313     def register_torpedo(self, torpedo):
314         """
315         This is called to register a new torpedo in our system
316
317         :param asteroid: This is your torpedo object
318         :type asteroid: Torpedo
319         """
320         if id(torpedo) in self._torpedos:
321             print("Error: Torpedo id (%d) already exists"%torpedo.id)
322             sys.exit(0)
323         torpedo_obj = self._get_torpedo_object()
324         self._torpedos[ id(torpedo) ] = torpedo_obj
325
326     def draw_ship(self, x, y, heading):
327         """
328         Draw the ship at the given coordinates with the given heading
329
330         :param x: This is the X coordinate of the ship
331         :type x: int

```



```

332         :param y: This is the Y coordinate of the ship
333         :type y: int
334         :param heading: This is the heading of the ship (in degrees)
335         :type heading: float
336
337         """
338         self._draw_object(self._ship, x, y, heading)
339
340     def draw_asteroid(self, asteroid, x, y):
341         """
342         Draw the given asteroid on the specified (x,y) coordinates
343
344         :param asteroid: This is your asteroid object (remember to register it before)
345         :type asteroid: Asteroid
346         :param x: This is the X coordinate of the asteroid
347         :type x: int
348         :param y: This is the Y coordinate of the asteroid
349         :type y: int
350
351         """
352         asteroid_id = id(asteroid)
353         if asteroid_id not in self._asteroids:
354             print("Error: Asteroid id (%d) not found. "%asteroid_id +
355                   "Are you sure there is such an asteroid?")
356             sys.exit(0)
357
358         self._draw_object(self._asteroids[asteroid_id], x, y)
359
360     def draw_torpedo(self, torpedo, x, y, heading):
361         """
362         Draw the given torpedo on the specified (x,y) coordinates with the given heading
363
364         :param asteroid: This is your torpedo object (remember to register it before)
365         :type asteroid: Torpedo
366         :param x: This is the X coordinate of the torpedo
367         :type x: int
368         :param y: This is the Y coordinate of the torpedo
369         :type y: int
370         :param heading: This is the heading of the torpedo
371         :type heading: float
372         """
373         torpedo_id = id(torpedo)
374         if torpedo_id not in self._torpedos:
375             print("Torpedo id (%d) not found. "%torpedo_id +
376                   "Are you sure there is such a torpedo?")
377             sys.exit(0)
378
379         self._draw_object(self._torpedos[torpedo_id], x, y, heading)
380
381     def _remove_object(self, obj):
382         obj.penup()
383         obj.ht()
384         obj.goto(Screen.SCREEN_MAX_X, Screen.SCREEN_MAX_Y*2)
385
386
387     def unregister_torpedo(self, torpedo):
388         """
389         This is called to un-register an existing torpedo in our system
390
391         :param asteroid: This is your torpedo object
392         :type asteroid: Torpedo
393         """
394         torpedo_id = id(torpedo)
395         if torpedo_id not in self._torpedos:
396             print("Torpedo id (%d) not found. "%torpedo_id +
397                   "Are you sure there is such a torpedo?")
398             sys.exit(0)
399         torpedo_obj = self._torpedos[torpedo_id]

```

```

400         self._remove_object( torpedo_obj )
401         self._torpedos.pop( torpedo_id )
402
403
404     def unregister_asteroid(self, asteroid):
405         """
406         This is called to un-register an existing asteroid in our system
407
408         :param asteroid: This is your asteroid object
409         :type asteroid: Asteroid
410         """
411         asteroid_id = id(asteroid)
412         if asteroid_id not in self._asteroids:
413             print("Asteroid id (%d) not found. "%asteroid_id +
414                   "Are you sure there is such an asteroid?")
415             sys.exit(0)
416         asteroid_obj = self._asteroids[ asteroid_id ]
417         self._remove_object( asteroid_obj )
418         self._asteroids.pop( asteroid_id )
419
420     def _clear_screen(self):
421         self._cv.delete('all')
422
423
424     def should_end(self):
425         """
426         :returns: True if the game should end or not (if "q" was pressed or not)
427         """
428         return self._endGame
429
430
431     def is_left_pressed(self):
432         """
433         :returns: True if the left key was pressed, else False
434         """
435         res = self._leftClicks > 0
436         self._leftClicks -= 1 if res else 0
437         return res
438
439     def is_up_pressed(self):
440         """
441         :returns: True if the up key was pressed, else False
442         """
443         res = self._upClicks > 0
444         self._upClicks -= 1 if res else 0
445         return res
446
447     def is_right_pressed(self):
448         """
449         :returns: True if the right key was pressed, else False
450         """
451         res = self._rightClicks > 0
452         self._rightClicks -= 1 if res else 0
453         return res
454
455     def is_space_pressed(self):
456         """
457         :returns: True if the fire key was pressed, else False
458         """
459         res = self._fireClicks > 0
460         self._fireClicks -= 1 if res else 0
461         return res
462
463     def is_special_pressed(self):
464         """
465         :returns: True if the fire key was pressed, else False
466         """
467         res = self._specialTorpedFired > 0

```

```

468         self._specialTorpedFired -= 1 if res else 0
469         return res
470
471     def show_message(self, title, msg):
472         """
473         This is a method used to show messages in the game.
474
475         :param title: The title of the message box.
476         :type title: str
477         :param msg: The message to show in the message box.
478         :type msg: str
479         """
480         tkinter.messagebox.showinfo(str(title), str(msg) )
481
482     def end_game(self):
483         """
484         This ends the current game.
485         """
486         self._root.destroy()
487         self._root.quit()

```

## 7 ship.py

```
1  from random import randrange
2  from screen import Screen as Scr
3  import copy
4  import math
5
6
7  class Ship:
8      LEFT_MOVE = 7
9      RIGHT_MOVE = -7
10     DELTA_X = Scr.SCREEN_MAX_X - Scr.SCREEN_MIN_X
11     DELTA_Y = Scr.SCREEN_MAX_Y - Scr.SCREEN_MIN_Y
12     RADIUS = 1
13
14     def __init__(self):
15         self.__x_speed = 0
16         self.__y_speed = 0
17         self.__x_position = randrange(Scr.SCREEN_MIN_X, Scr.SCREEN_MAX_X)
18         self.__y_position = randrange(Scr.SCREEN_MIN_Y, Scr.SCREEN_MAX_Y)
19         self.__heading = 0
20         self.__life = 3
21
22     def get_x_position(self):
23         return copy.copy(self.__x_position)
24
25     def get_y_position(self):
26         return copy.copy(self.__y_position)
27
28     def get_x_speed(self):
29         return copy.copy(self.__x_speed)
30
31     def get_y_speed(self):
32         return copy.copy(self.__y_speed)
33
34     def get_heading(self):
35         return copy.copy(self.__heading)
36
37     def get_life(self):
38         return copy.copy(self.__life)
39
40     def change_heading_left(self):
41         """
42         This function updates left movement for the spaceship.
43         """
44         self.__heading += self.LEFT_MOVE
45         return copy.copy(self.__heading)
46
47     def change_heading_right(self):
48         """
49         This function updates right movement for the spaceship.
50         """
51         self.__heading += self.RIGHT_MOVE
52         return copy.copy(self.__heading)
53
54     def accelerated_ship(self):
55         """
56         This function defines the x and y of the ship when we want our ship
57         to accelerate.
58         """
59         radian_heading = math.radians(self.__heading)
```

```

60         self.__x_speed += math.cos(radian_heading)
61         self.__y_speed += math.sin(radian_heading)
62         return self.__x_speed, self.__y_speed
63
64     def move(self):
65         """
66         This function calculates our new coordinate of the position for our
67         ship.
68         """
69         self.__x_position = ((self.get_x_speed() + self.get_x_position() -
70                               Scr.SCREEN_MIN_X) % self.DELTA_X + Scr.SCREEN_MIN_X)
71         self.__y_position = ((self.get_y_speed() + self.get_y_position() -
72                               Scr.SCREEN_MIN_Y) % self.DELTA_Y + Scr.SCREEN_MIN_Y)
73
74     def update_life(self):
75         """
76         This function updates the amount of lives that the ship has left by
77         reducing one of the remaining lives.
78         """
79         self.__life -= 1

```

## 8 torpedo.py

```
1 import copy
2 import math
3 from screen import Screen as Scr
4
5 class Torpedo:
6
7     """
8     Torpedo is a class mainly based on the class Ship's data. Because the
9     torpedo is derived from the Ship, it's position and speed are attributes
10    from which are dependant on the ships position and speed.
11    """
12
13    RADIUS = 4
14    ACCELERATING_FACTOR = 2
15    DELTA_X = Scr.SCREEN_MAX_X - Scr.SCREEN_MIN_X
16    DELTA_Y = Scr.SCREEN_MAX_Y - Scr.SCREEN_MIN_Y
17    LIFESPAN_MAX = 200
18
19    def __init__(self, x_pos ,y_pos, x_speed, y_speed, direction):
20        self.__x_position = x_pos
21        self.__y_position = y_pos
22        self.__x_speed = x_speed
23        self.__y_speed = y_speed
24        self.__direction = direction
25        self.__lifespan = 1
26        self.find_speed()
27
28    def get_x_position(self):
29        return copy.copy(self.__x_position)
30
31    def get_y_position(self):
32        return copy.copy(self.__y_position)
33
34    def get_x_speed(self):
35        return copy.copy(self.__x_speed)
36
37    def get_y_speed(self):
38        return copy.copy(self.__y_speed)
39
40    def get_direction(self):
41        return copy.copy(self.__direction)
42
43    def get_lifespan(self):
44        return copy.copy(self.__lifespan)
45
46    def find_speed(self):
47        """
48        This function calculates the speed of the torpedo.
49        """
50        radian_direction = math.radians(self.__direction)
51        self.__x_speed += self.ACCELERATING_FACTOR * math.cos(radian_direction)
52        self.__y_speed += self.ACCELERATING_FACTOR * math.sin(radian_direction)
53
54    def update_lifespan(self):
55        """
56        This function "clocks" the lifespan by adding 1 each time the function
57        is called.
58        """
59        self.__lifespan += 1
```

```

60
61     def move(self):
62         """
63         This function calculates our new coordinate of the position for our
64         ship.
65         """
66         self.__x_position = ((self.get_x_speed() + self.get_x_position() -
67                               Scr.SCREEN_MIN_X) % self.DELTA_X + Scr.SCREEN_MIN_X)
68         self.__y_position = ((self.get_y_speed() + self.get_y_position() -
69                               Scr.SCREEN_MIN_Y) % self.DELTA_Y + Scr.SCREEN_MIN_Y)

```