```
#Some of the code below is taken from another game
#Link to game: [https://trinket.io/python/4fb8b43036]

from turtle import *
import turtle
import random
import math

# sets screen resolution
screen = Screen()
screen.setworldcoordinates(-230,-230,230,230)
screen.bgcolor("black")

initialSetup="true"
score=0

# define lists for asteroids and bullets
asteroids = []
bullets = []

#registers the shape that the asteroids takes
screen.register_shape("emote.png", turtle.circle(10))

# registering the object of bullet to the screen itself
screen.register_shape("bullet",((-2,-4),(-2,4),(2,4),(2,-4)))

hideturtle()

#  defining the basic functions of the scoreboard turtle
scoreboard = turtle.Turtle()
scoreboard.hideturtle()
scoreboard.penup()
scoreboard.goto(-217,208)

scoreboard.color("red")
scoreString = "Score: " + str(score)
scoreboard.write(scoreString, None, "left", font=('Arial', 16, 'normal'))


# defines what bullet will do, shape and color and direction
class bullet(Turtle):
  def __init__(self,screen,x,y,heading):
    Turtle.__init__(self)
    self.screen = screen
    self.seth(heading)
    self.shape("bullet")
    self.color('red')
    self.penup()
    self.goto(x,y)

  # defines how a bullet will move
  def move(self):
    global bulletDist
    bulletDist=0
    bulletDist = bulletDist + 20
    self.forward(20)
```

```
57      if bulletDist >= 600:
58        self.reset()
59
60    def hitbox(self):
61      return 6
62
63    def boom(self):
64      self.goto(-420,0)
65
66  # defines what asteroid will do, picture of asteroid, speed that asteroid will move
67  class Asteroid(Turtle):
68    def __init__(self,screen,dx,dy,x,y,size):
69      Turtle.__init__(self)
70      self.speed(0)
71      self.penup()
72      self.goto(x,y)
73      self.size = size
74      self.screen = screen
75      self.dx = dx
76      self.dy = dy
77      self.color('lightgrey')
78      self.shape("emote.png")
79
80    def move(self):
81      x = self.xcor()
82      y = self.ycor()
83
84      x = (self.dx + x +230) % 460 + -230
85      y = (self.dy + y +230) % 460 + -230
86
87      self.goto(x,y)
88
89    def boom(self):
90      self.goto(-420,0)
91
92    def hitbox(self):
93      return self.size * 20
94
95  # defines rocket, color, speed and movement
96  class rocket(Turtle):
97    def __init__(self,screen,dx,dy,x,y):
98      Turtle.__init__(self)
99      self.screen = screen
100     self.bullets = []
101     self.speed(0)
102     self.color("white")
103     self.dx = dx
104     self.dy = dy
105     self.penup()
106     self.goto(x,y)
107
108   def move(self):
109     x = self.xcor()
110     y = self.ycor()
111
112     x = (self.dx + x +230) % 460 + -230
```

```
113       y = (self.dy + y +230) % 460 + -230
114
115       self.goto(x,y)
116
117     #this code is used to obtain the exact angle and direction(steering) of the rocket
118     #this code is from [https://trinket.io/python/4fb8b43036]
119     def fireEngine(self):
120       angle = self.heading()
121       x = math.cos(math.radians(angle))
122       y = math.sin(math.radians(angle))
123       self.dx = self.dx + x
124       self.dy = self.dy + y
125
126     # defines interaction when bullet hits with asteroid, score will increase and be
127     # added up, asteroid that is hit by bullet will be removed from screen
128     def bulletHit(self, asteroids):
129       bulletList = []
130       global score
131       for bullet in self.bullets:
132         bullet.move()
133         hit = False
134         for asteroid in asteroids:
135           if collide(asteroid, bullet):
136             asteroids.remove(asteroid)
137             asteroid.boom()
138             bullet.boom()
139
140             # score variable which adds 1 point for every asteroid collided with bullet
141             score=score+1
142             scoreboard.clear()
143             scoreString = "Score: " + str(score)
144             scoreboard.write(scoreString, None, "left", font=('Arial', 16, 'normal'))
145
146            # modifies generation of astroids after scores 60+ and 100+
147             if (score >= 100):
148               asteroidGenerate(15,0)
149             elif (score >=60):
150               asteroidGenerate(5,1)
151             else:
152               asteroidGenerate(2,1)
153
154             hit = True
155
156         # creates a list of only active bullets left
157         if (not bullet.done() and not hit):
158           bulletList.append(bullet)
159
160
161       self.bullets = bulletList
162
163   # function that fires a bullet and creates a new bullet
164     def fireBullet(self):
165       self.bullets.append(bullet(self.screen, self.xcor(), self.ycor(), self.heading()))
166
167     def hitbox(self):
168         return 6
```

```
169
170 # defines collision of objects, if the radius of two objects combined is greater or
171 # equal to the distance between objects then the two objects are considered collided
172 def collide(object1,object2):
173
174   dist = math.sqrt((object1.xcor() - object2.xcor())**2 + (object1.ycor() - object2.ycor())**2)
175
176   radius1 = object1.hitbox()
177   radius2 = object2.hitbox()
178
179   if dist <= radius1+radius2:
180       return True
181   else:
182       return False
183
184 ship = rocket(screen,0,0,0,0)
185
186 # defines asteroid generation when game starts,
187 # makes sure asteroid does not spawn right on top of rocket spawn area
188 def asteroidGenerate(speed,enableSpawnCheck):
189
190
191   if (enableSpawnCheck==1):
192     spawnCheck="unsafe"
193     while (spawnCheck=="unsafe"):
194       dx = random.random() * 6 - speed
195       dy = random.random() * 6 - speed
196       x = random.random() * 460 + 230
197       y = random.random() * 460 + 230
198
199
200       spawnDist = math.sqrt((x - ship.xcor())**2 + (x - ship.ycor())**2)
201
202       if spawnDist >= 100:
203         spawnCheck="safe"
204       else:
205         spawnCheck="unsafe"
206
207     asteroid = Asteroid(screen,dx,dy,x,y, 2 )
208     spawnCheck="unsafe"
209     asteroids.append(asteroid)
210
211   else:
212     dx = random.random() * 6 - speed
213     dy = random.random() * 6 - speed
214     x = random.random() * 460 + 230
215     y = random.random() * 460 + 230
216
217     asteroid = Asteroid(screen,dx,dy,x,y, 2)
218     asteroids.append(asteroid)
219
220 if (initialSetup=="true"):
221   for i in range(6):
222     asteroidGenerate(2,1)
223   initialSetup="false"
224
```

```
225 # defines what happen when your ship moves into asteroid,
226 # game will end and display "final score" in top left and "BamBam!" in the center
227 def gameplay():
228
229   ship.move()
230
231   gameover = False
232   for asteroid in asteroids:
233     asteroid.move()
234     if collide(ship,asteroid):
235       scoreboard.goto(-217,208)
236       scoreboard.clear()
237       scoreString = "Final Score: " + str(score)
238       scoreboard.write(scoreString, None, "left", font=('Arial', 16, 'normal'))
239
240       color("red")
241       write("BamBam!",font=("Arial",24),align="center" )
242       print "Final Score: ", score
243       gameover = True
244
245   ship.bulletHit(asteroids)
246   screen.update()
247
248   # defines when game is not over and how the ship moves
249   if not gameover:
250     screen.ontimer(gameplay, 30)
251
252 def turnLeft():
253   ship.left(7)
254
255 def turnRight():
256   ship.right(7)
257
258 screen.tracer(0);
259
260 # Inputs to outputs, when using left and arrow, will turn rocket in respective direction
261 # When using up key, will move rocket forward.
262 # When using space key, will cause ship to fire bullets
263 screen.onkey(turnLeft, 'left')
264 screen.onkey(turnRight, 'right')
265 screen.onkey(ship.fireEngine, 'up')
266 screen.onkey(ship.fireBullet, 'space')
267 screen.listen()
268
269 gameplay()
```