```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
1   # Snake Heart
2   # Code Angel
3
4   # Classes: Monster, Bumber, Whizzer and Boxer
5
6   import utils
7
8   import screen
9   import random
10
11
12  class Monster:
13
14      def __init__(self, image, speed):
15
16          self.monster_image = utils.load_media('image', image)
17
18          self.rect = self.monster image.get rect()
19          self.width = self.rect.width
20          self.height = self.rect.height
21          self.speed = speed
22
23          self.direction = None
24
25          self.alive = True
26
27          # The terrain types at either side of the monster
28          self.terrain_type_1 = None
29          self.terrain_type_2 = None
30          self.terrain_type_3 = None
31          self.terrain_type_4 = None
32
33      def standard_direction(self):
34          self.direction = random.choice(['left', 'right', 'up', 'down'])
35
36      def diagonal_direction(self):
37          self.direction = random.choice(['up right', 'down right', 'down left', 'up left'])
38
39      # Calculate a random spawn location which must be on land
40      def spawn_location(self, game_map):
41
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
42          start_tile_col = random.randint(0, game_map.map_cols - 1)
43          start_tile_row = random.randint(0, game_map.map_rows - 1)
44
45          terrain = game_map.map_key.get(game_map.tile_list[start_tile_row][start_tile_col])
46          while terrain != 'land':
47              start_tile_col = random.randint(0, game_map.map_cols - 1)
48              start_tile_row = random.randint(0, game_map.map_rows - 1)
49
50              terrain = game_map.map_key.get(game_map.tile_list[start_tile_row][start_tile_col])
51
52          self.rect.x = (start_tile_col - game_map.map_tile_x) * screen.TILE_SIZE
53          self.rect.y = (start_tile_row - game_map.map_tile_y) * screen.TILE_SIZE
54
55      # Draw the monster passing the image and location to the Display class
56      def draw(self, display):
57          if self.alive is True:
58              display.show_image(self.monster_image, self.rect.x, self.rect.y)
59
60      # Move the monster
61      def move(self, game_map):
62
63          if self.alive is True:
64
65              # Update the x and y location based on the direction the monster is moving in
66              if 'right' in self.direction:
67                  self.rect.x += self.speed
68              if 'left' in self.direction:
69                  self.rect.x -= self.speed
70              if 'down' in self.direction:
71                  self.rect.y += self.speed
72              if 'up' in self.direction:
73                  self.rect.y -= self.speed
74
75              # Update the monster's location based on any map scrolling
76              self.rect.x += game_map.dx
77              self.rect.y += game_map.dy
78
79              # Update the surrounding terrain details, check if fallen into hole / drowned, and maybe change
80  direction
81              self.update_surrounding_terrain(game_map)
82              self.check_for_hole()
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```



```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
83              self.check_if_drowned()
84              self.maybe_change_direction()
85
86      # The tiles surrounding the monster
87      # Used because the map scrolls by less than a full tile size
88      def update_surrounding_terrain(self, game_map):
89          monster_tile_x_1 = int(self.rect.x / screen.TILE_SIZE) + 1
90          monster_tile_x_2 = int((self.rect.x + self.rect.width) / screen.TILE_SIZE) + 1
91          monster_tile_y_1 = int(self.rect.y / screen.TILE_SIZE) + 1
92          monster_tile_y_2 = int((self.rect.y + self.rect.height) / screen.TILE_SIZE) + 1
93
94          # Work out what is actually in the terrain surrounding the monster
95          rel_y1 = game_map.map_tile_y + monster_tile_y_1
96          rel_x1 = game_map.map_tile_x + monster_tile_x_1
97          rel_y2 = game_map.map_tile_y + monster_tile_y_2
98          rel_x2 = game_map.map_tile_x + monster_tile_x_2
99          self.terrain_type_1 = game_map.map_key.get(game_map.tile_list[rel_y1][rel_x1])
100         self.terrain_type_2 = game_map.map_key.get(game_map.tile_list[rel_y1][rel_x2])
101         self.terrain_type_3 = game_map.map_key.get(game_map.tile_list[rel_y2][rel_x1])
102         self.terrain_type_4 = game_map.map_key.get(game_map.tile_list[rel_y2][rel_x2])
103
104     # Test if the monster fallen into a hole
105     def check_for_hole(self):
106
107         if self.alive is True:
108
109             if (self.terrain_type_1 == 'trap' or
110                     self.terrain_type_2 == 'trap' or
111                     self.terrain_type_3 == 'trap' or
112                     self.terrain_type_4 == 'trap'):
113
114                 # The monster has fallen into a hole, so alive will be false and kills need to be updated
115                 self.alive = False
116                 self.update_kills()
117
118     # Overridden in Bumbler, Whizzer and Boxer
119     def update_kills(self):
120         pass
121
122     # Test if the monster has fallen into water
123     def check_if_drowned(self):
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
124         if self.alive is True:
125             if (self.terrain_type_1 == 'water' or
126                     self.terrain_type_2 == 'water' or
127                     self.terrain_type_3 == 'water' or
128                     self.terrain_type_4 == 'water'):
129
130                 self.alive = False
131
132     # Overridden in Bumbler and Boxer - Whizzer does not change direction
133     def maybe_change_direction(self):
134         pass
135
136     # The player has used a portal, so the relative postion of the monster will need to change accordingly
137     def portal(self, portal_x, portal_y):
138         self.rect.x += portal_x * screen.TILE_SIZE
139         self.rect.y += portal_y * screen.TILE_SIZE
140
141
142 # Bumbler class inherits from Monster
143 class Bumbler(Monster):
144
145     # Class variables
146
147     # Chance of a Bumbler being spawned
148     spawn_chance = 200
149
150     # Number of Bumblers killed
151     kills = 0
152
153     # Maximum number of Bumblers in the game
154     max = 150
155
156     def __init__(self, level):
157
158         speed = 0
159         self.random_direction = 0
160         self.intelligence = 0
161
162         if level == 1:
163             self.random_direction = 30
164             self.intelligence = 5
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
165             speed = 1
166         elif level == 2:
167             self.random_direction = 20
168             self.intelligence = 3
169             speed = 2
170
171         super().__init__('monster_down', speed)
172         super().standard_direction()
173
174     # Static method to spawn a bumbler
175     @staticmethod
176     def spawn(bumblers, game_map):
177         random_spawn = random.randint(1, Bumbler.spawn_chance)
178         if random_spawn == 1 and len(bumblers) < Bumbler.max:
179             bumbler = Bumbler(game_map.level)
180             bumbler.spawn_location(game_map)
181             bumblers.append(bumbler)
182
183     # Increase the class variable kills
184     def update_kills(self):
185         Bumbler.kills += 1
186
187     # Randomly generate a change in direction
188     def maybe_change_direction(self):
189
190         # chance change is the chance of changing direction
191         # The higher random_direction, the less likely a random change
192         chance_change = random.randint(1, self.random_direction)
193
194         # Only change if the randomly generated value is 1
195         if chance_change == 1:
196
197             # smart change means going in the direction of player
198             # lower intelligence number, greater chance of a smart change rather than random
199             smart_change = random.randint(1, self.intelligence)
200
201             if smart_change == 1:
202
203                 # The player is always at the centre of the map
204                 # Move vertical direction towards player
205                 if self.rect.y < screen.SCREENHEIGHT / 2:
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
206                     self.direction = 'down'
207                 else:
208                     self.direction = 'up'
209             elif smart_change == 2:
210
211                 # Move horizontal direction towards player
212                 if self.rect.x < screen.SCREENWIDTH / 2:
213                     self.direction = 'right'
214                 else:
215                     self.direction = 'left'
216
217
218 # Whizzer class inherits from Monster
219 class Whizzer(Monster):
220
221     spawn_chance = 50
222     kills = 0
223     max = 75
224
225     def __init__(self, level):
226         speed = 0
227
228         if level == 1:
229             speed = 6
230         elif level == 2:
231             speed = 10
232
233         super().__init__('whizzer_down', speed)
234         super().standard_direction()
235
236     @staticmethod
237     def spawn(whizzers, game_map):
238         random_spawn = random.randint(1, Whizzer.spawn_chance)
239         if random_spawn == 1 and len(whizzers) < Whizzer.max:
240             whizzer = Whizzer(game_map.level)
241             whizzer.spawn_location(game_map)
242             whizzers.append(whizzer)
243
244     def update_kills(self):
245         Whizzer.kills += 1
246
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
247
248 # Boxer class inherits from Monster
249 class Boxer(Monster):
250
251     spawn_chance = 200
252     kills = 0
253     max = 100
254
255     def __init__(self, level):
256         self.random_direction = 0
257         speed = 0
258
259         if level == 1:
260             self.random_direction = 15
261             speed = 3
262         elif level == 2:
263             self.random_direction = 5
264             speed = 5
265
266         super().__init__('boxer_down', speed)
267         super().diagonal_direction()
268
269     def maybe_change_direction(self):
270
271         chance_change = random.randint(1, self.random_direction)
272
273         if chance_change == 1:
274             if self.direction == 'up right':
275                 self.direction = 'down right'
276             elif self.direction == 'down right':
277                 self.direction = 'down left'
278             elif self.direction == 'down left':
279                 self.direction = 'up left'
280             else:
281                 self.direction = 'up right'
282
283     @staticmethod
284     def spawn(boxers, game_map):
285         random_spawn = random.randint(1, Boxer.spawn_chance)
286         if random_spawn == 1 and len(boxers) < Boxer.max:
287             boxer = Boxer(game_map.level)
    |0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
    |0  |1  |2  |3  |4  |5  |6  |7  |8
288             boxer.spawn_location(game_map)
289             boxers.append(boxer)
290
291     def update_kills(self):
292         Boxer.kills += 1
293
```

|0 |1 |2 |3 |4 |5 |6 |7 |8

