```
|0   |1   |2   |3   |4   |5   |6   |7   |8
# Snake Heart
# Code Angel

# Classes: Map

import pygame
import screen
import csv
import random
import os

import player_class
import utils


# Map class
class Map:

    def __init__(self):

        # Load all map images
        self.water_image = utils.load_media('image', 'water')
        self.land_image = utils.load_media('image', 'land')
        self.portal_image = utils.load_media('image', 'portal')
        self.re_port_image = utils.load_media('image', 're-port')
        self.beach_image = utils.load_media('image', 'beach')
        self.gold_image = utils.load_media('image', 'gold')
        self.trap_image = utils.load_media('image', 'hole')
        self.heart_image = utils.load_media('image', 'heart')
        self.spade_image = utils.load_media('image', 'spade')

        self.sword_1_image = utils.load_media('image', 'sword_1')
        self.sword_2_image = utils.load_media('image', 'sword_2')
        self.sword_3_image = utils.load_media('image', 'sword_3')
        self.sword_4_image = utils.load_media('image', 'sword_4')

        self.castle_1_image = utils.load_media('image', 'castle_1')
        self.castle_2_image = utils.load_media('image', 'castle_2')
        self.castle_3_image = utils.load_media('image', 'castle_3')
        self.castle_4_image = utils.load_media('image', 'castle_4')
        self.castle_5_image = utils.load_media('image', 'castle_5')
|0   |1   |2   |3   |4   |5   |6   |7   |8
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
        self.castle_6_image = utils.load_media('image', 'castle_6')

        # Load all map audio
        self.gold_sound = utils.load_media('audio', 'gold')
        self.extra_life_sound = utils.load_media('audio', 'extra_life')
        self.spade_sound = utils.load_media('audio', 'spade')
        self.sword_sound = utils.load_media('audio', 'sword')
        self.portal_sound = utils.load_media('audio', 'portal')
        self.water_sound = utils.load_media('audio', 'water')

        self.map_key = {
            'w': 'water',
            'l': 'land',
            'p': 'portal',
            'r': 're-port',
            'b': 'beach',
            'g': 'gold',
            't': 'trap',
            'h': 'heart',
            'd': 'spade',
            '1': 'sword 1',
            '2': 'sword 2',
            '3': 'sword 3',
            '4': 'sword 4',
            'c1': 'castle 1',
            'c2': 'castle 2',
            'c3': 'castle 3',
            'c4': 'castle 4',
            'c5': 'castle 5',
            'c6': 'castle 6'
        }

        # Total number of rows and columns in the map
        self.screen_cols = int(screen.SCREENWIDTH / screen.TILE_SIZE)
        self.screen_rows = int(screen.SCREENHEIGHT / screen.TILE_SIZE)

        self.map_cols = None
        self.map_rows = None

        self.re_port_points = []

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

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
        self.map_tile_x = 0
        self.map_tile_y = 0
        self.map_tile_step_x = 0
        self.map_tile_step_y = 0

        self.player_row = int(screen.SCREENHEIGHT / screen.TILE_SIZE / 2 + 1)
        self.player_col_left = int(screen.SCREENWIDTH / screen.TILE_SIZE / 2)
        self.player_col_right = int(screen.SCREENWIDTH / screen.TILE_SIZE / 2 + 1)

        self.dx = 0
        self.dy = 0

        self.portal = False
        self.d_portal_x = 0
        self.d_portal_y = 0

        self.level = 0
        self.tile_list = None

    # Load the CSV map
    def load_map(self):

        map_file = None

        if self.level == 1:
            map_file = 'snakeheart map level 1'
        elif self.level == 2:
            map_file = 'snakeheart map level 2'

        full_path = os.path.dirname(os.path.realpath(__file__))
        maps_path = os.path.join(full_path, 'maps')
        full_filename = os.path.join(maps_path, map_file + '.csv')

        # Open and read the map CSV file and store in 2D list tile list
        with open(full_filename, 'r') as csvfile:
            read_map = csv.reader(csvfile)
            self.tile_list = list(read_map)

        # Calculate total map rows and columns
        self.map_cols = len(self.tile_list[0])
        self.map_rows = len(self.tile_list)
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
        # Find all map re-port points
        self.find_re_port_points()

    # Scroll map as player moves
    def scroll(self, direction):

        self.dx = 0
        self.dy = 0

        if direction == 'right':
            if self.map_tile_x > 0:
                self.dx = player_class.PLAYER_MOVE
                self.dy = 0

        elif direction == 'left':
            if self.map_tile_x < self.map_cols - self.screen_cols - 1:
                self.dx = -player_class.PLAYER_MOVE
                self.dy = 0

        elif direction == 'down':
            if self.map_tile_y > 0:
                self.dx = 0
                self.dy = player_class.PLAYER_MOVE

        elif direction == 'up':
            if self.map_tile_y < self.map_rows - self.screen_rows - 1:
                self.dx = 0
                self.dy = -player_class.PLAYER_MOVE

        # Work out if a complete tile has been scrolled, and if so update map_tile_x / map_tile_y
        self.map_tile_step_x += self.dx
        self.map_tile_step_y += self.dy

        if self.map_tile_step_x >= screen.TILE_SIZE:
            self.map_tile_step_x -= screen.TILE_SIZE
            self.map_tile_x -= 1

        if self.map_tile_step_x < 0:
            self.map_tile_step_x += screen.TILE_SIZE
            self.map_tile_x += 1
```

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

        if self.map_tile_step_y >= screen.TILE_SIZE:
            self.map_tile_step_y -= screen.TILE_SIZE
            self.map_tile_y -= 1

        if self.map_tile_step_y < 0:
            self.map_tile_step_y += screen.TILE_SIZE
            self.map_tile_y += 1


    # Draw the map items
    def draw(self, display):
        for row in range(self.screen_rows + 1):
            for col in range(self.screen_cols + 1):

                lucy_row = row + self.map_tile_y
                lucy_col = col + self.map_tile_x
                tile_key = self.tile_list[lucy_row][lucy_col]

                tile = self.map_key.get(tile_key)

                display_image = None

                if tile == 'water':
                    display_image = self.water_image
                elif tile == 'land':
                    display_image = self.land_image
                elif tile == 'portal':
                    display_image = self.portal_image
                elif tile == 're-port':
                    display_image = self.re_port_image
                elif tile == 'gold':
                    display_image = self.gold_image
                elif tile == 'beach':
                    display_image = self.beach_image
                elif tile == 'trap':
                    display_image = self.trap_image
                elif tile == 'heart':
                    display_image = self.heart_image
                elif tile == 'spade':
                    display_image = self.spade_image
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
                elif tile == 'sword 1':
                    display_image = self.sword_1_image
                elif tile == 'sword 2':
                    display_image = self.sword_2_image
                elif tile == 'sword 3':
                    display_image = self.sword_3_image
                elif tile == 'sword 4':
                    display_image = self.sword_4_image

                elif tile == 'castle 1':
                    display_image = self.castle_1_image
                elif tile == 'castle 2':
                    display_image = self.castle_2_image
                elif tile == 'castle 3':
                    display_image = self.castle_3_image
                elif tile == 'castle 4':
                    display_image = self.castle_4_image
                elif tile == 'castle 5':
                    display_image = self.castle_5_image
                elif tile == 'castle 6':
                    display_image = self.castle_6_image

                x_pos = (col - 1) * screen.TILE_SIZE + self.map_tile_step_x
                y_pos = (row - 1) * screen.TILE_SIZE + self.map_tile_step_y
                display.show_image(display_image, x_pos, y_pos)

    def reset_change(self):
        self.dx = 0
        self.dy = 0

    # Test if Lucy has come into contact with any map items
    def check_player_loc(self, player):
        player_x = self.player_row + self.map_tile_y + 1
        player_y_left = self.player_col_left + self.map_tile_x
        player_y_right = self.player_col_right + self.map_tile_x

        tile_left = self.map_key.get(self.tile_list[player_x][player_y_left])
        tile_right = self.map_key.get(self.tile_list[player_x][player_y_right])

        touching_water = self.player_touching('water', tile_left, tile_right)
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
        touching_gold = self.player_touching('gold', tile_left, tile_right)
        touching_portal = self.player_touching('portal', tile_left, tile_right)
        touching_heart = self.player_touching('heart', tile_left, tile_right)
        touching_spade = self.player_touching('spade', tile_left, tile_right)

        touching_sword_1 = self.player_touching('sword 1', tile_left, tile_right)
        touching_sword_2 = self.player_touching('sword 2', tile_left, tile_right)
        touching_sword_3 = self.player_touching('sword 3', tile_left, tile_right)
        touching_sword_4 = self.player_touching('sword 4', tile_left, tile_right)

        touching_castle_1 = self.player_touching('castle 1', tile_left, tile_right)
        touching_castle_2 = self.player_touching('castle 2', tile_left, tile_right)
        touching_castle_3 = self.player_touching('castle 3', tile_left, tile_right)
        touching_castle_4 = self.player_touching('castle 4', tile_left, tile_right)
        touching_castle_5 = self.player_touching('castle 5', tile_left, tile_right)
        touching_castle_6 = self.player_touching('castle 6', tile_left, tile_right)

        if touching_water is True and player.alive is True:
            player.map_water()
            self.water_sound.play()

        elif touching_gold is True:
            player.map_gold()
            self.remove_item('gold', tile_left, tile_right)
            self.gold_sound.play()

        elif touching_heart is True:
            if player.lives < player.max_lives:
                player.map_heart()
                self.remove_item('heart', tile_left, tile_right)
                self.extra_life_sound.play()

        elif touching_spade is True:
            if player.spades < player.max_spades:
                player.map_spade()
                self.remove_item('spade', tile_left, tile_right)
                self.spade_sound.play()

        elif touching_portal is True:
            self.portal_move()
            self.portal_sound.play()
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

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

        elif touching_sword_1 is True:
            player.map_sword(1)
            self.remove_item('sword 1', tile_left, tile_right)
            self.sword_sound.play()

        elif touching_sword_2 is True:
            player.map_sword(2)
            self.remove_item('sword 2', tile_left, tile_right)
            self.sword_sound.play()

        elif touching_sword_3 is True:
            player.map_sword(3)
            self.remove_item('sword 3', tile_left, tile_right)
            self.sword_sound.play()

        elif touching_sword_4 is True:
            player.map_sword(4)
            self.remove_item('sword 4', tile_left, tile_right)
            self.sword_sound.play()

        elif touching_castle_1 is True:
            player.map_castle()
        elif touching_castle_2 is True:
            player.map_castle()
        elif touching_castle_3 is True:
            player.map_castle()
        elif touching_castle_4 is True:
            player.map_castle()
        elif touching_castle_5 is True:
            player.map_castle()
        elif touching_castle_6 is True:
            player.map_castle()

    # Lucy has dug a hole / trap so add this 2 the map (6 pieces)
    def add_trap(self):
        self.add_trap_piece(self.player_row + self.map_tile_y, self.player_col_left + self.map_tile_x)
        self.add_trap_piece(self.player_row + self.map_tile_y, self.player_col_left + self.map_tile_x + 1)
        self.add_trap_piece(self.player_row + self.map_tile_y, self.player_col_left + self.map_tile_x + 2)
        self.add_trap_piece(self.player_row + self.map_tile_y + 1, self.player_col_left + self.map_tile_x)
        self.add_trap_piece(self.player_row + self.map_tile_y + 1, self.player_col_left + self.map_tile_x + 1)
|0  |1  |2  |3  |4  |5  |6  |7  |8
```

```
        self.add_trap_piece(self.player_row + self.map_tile_y + 1, self.player_col_left + self.map_tile_x + 2)

    # Add a trap piece ('t') only if the space is land ('l')
    def add_trap_piece(self, row_tile, col_tile):
        if self.tile_list[row_tile][col_tile] == 'l':
            self.tile_list[row_tile][col_tile] = 't'

    # Port to a random re-port point
    def portal_move(self):
        random_re_port = random.choice(self.re_port_points)
        new_map_x = random_re_port[1] - 20
        new_map_y = random_re_port[0] - 15 - 2

        self.d_portal_x = self.map_tile_x - new_map_x
        self.d_portal_y = self.map_tile_y - new_map_y
        self.map_tile_x = new_map_x
        self.map_tile_y = new_map_y
        self.map_tile_step_x = 0
        self.map_tile_step_y = 0

        self.portal = True

    # Find all the re-port points in the map and add to the list re_port_points
    def find_re_port_points(self):
        for row in range(self.map_rows):
            for col in range(self.map_cols):
                tile = self.map_key.get(self.tile_list[row][col])

                if tile == 're-port':
                    self.re_port_points.append([row, col])

    # Is the player touching an item
    def player_touching(self, item, tile_left, tile_right):
        player_touching_item = False
        if (tile_left == item and self.map_tile_step_x > player_class.PLAYER_MOVE) or tile_right == item:
            player_touching_item = True

        return player_touching_item

    # Remove an item by changing its map value to land ('l')
    def remove_item(self, item, tile_left, tile_right):
```

```
|0  |1  |2  |3  |4  |5  |6  |7  |8
        if tile_left == item and self.map_tile_step_x > player_class.PLAYER_MOVE:
            self.tile_list[self.player_row + self.map_tile_y + 1][self.player_col_left + self.map_tile_x] = 'l'
        elif tile_right == item:
            self.tile_list[self.player_row + self.map_tile_y + 1][self.player_col_right + self.map_tile_x] = 'l'

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