CS101 Homework #2

Snake 101

Due date: Sunday, April 12, 2015 (until 23:59)

Delayed Due date: Wednesday, April 15, 2015 (until 23:59)

Please read the homework description carefully and make sure that your program meets all the requirements stated. The homework is an individual task. You can discuss the problem with your friend, but you should not program together. **You will get an F on the entire course if your homework includes any plagiarism.**

Goal

Have you played a game "Snake"? If you have not played it yet, then you should have a look on some of its variants from the following link: http://playsnake.org/. Its goal is simple: to lengthen the snake as long as you can, by eating the objects while dodging the walls and its own body.

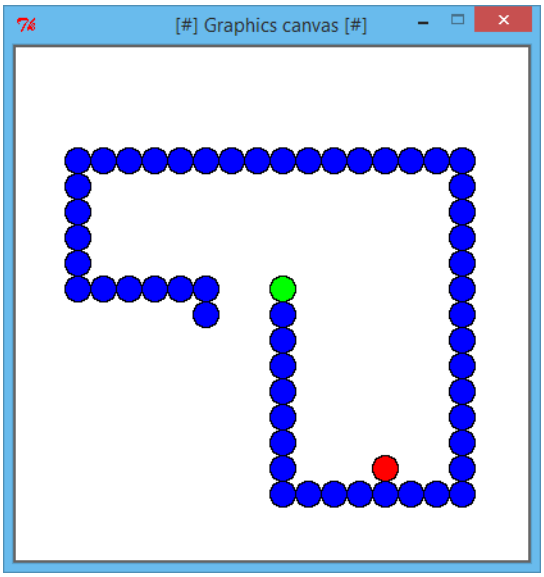

Figure 1 The Snake Game

In this homework, you will implement this game using cs1graphics. For your information, check the reference program (CS101_2015S_HW2_Reference.exe) to get a grasp of how it works. Use WASD keys to change the direction, and R key to restart the game.

Requirements

Your template file is given as **`HW2_template.py`**. Rename it to 'HW2_your_student_ID.py' (e.g., **'HW2_20151234.py'** if your student ID is 20151234) and modify it to implement the game. In total, there are 15 functions in the template source code and you are going to implement 8 functions among them. Please, read carefully the source code including global variables, call structure and existing function codes such as `main()`, `game_step()` and `draw_board()` functions, so that you may grab the overall structure of the program.

The game constants are the global variables that define the look and feel, as well as the difficulty of the game.

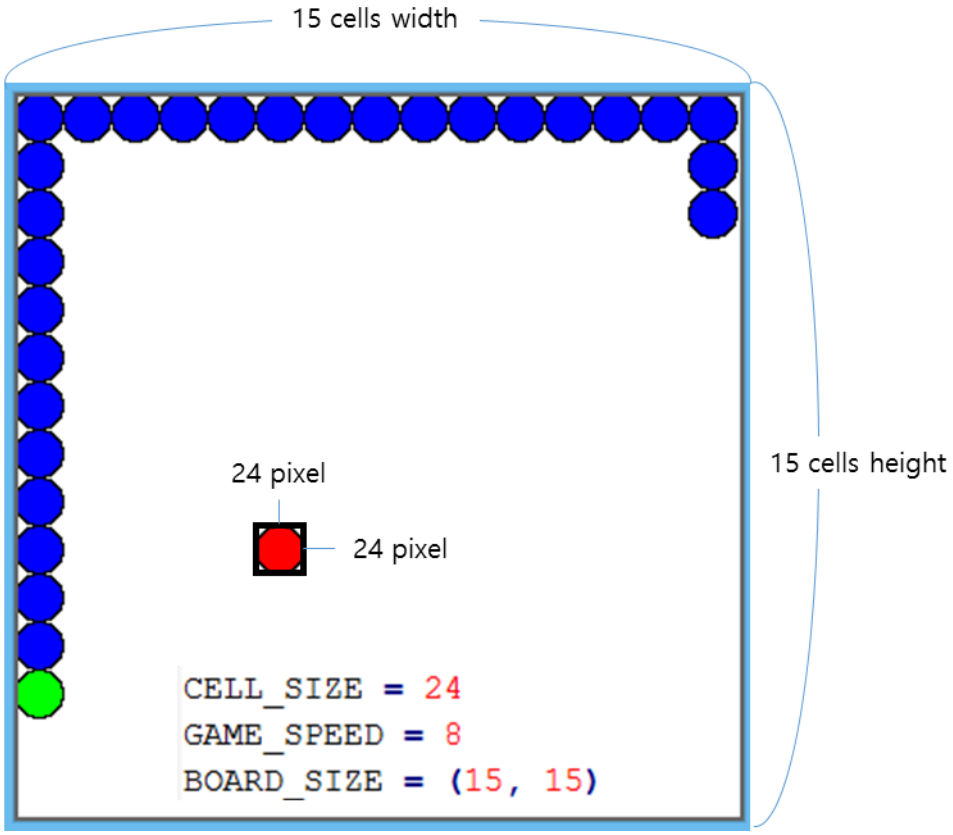

Figure 2 Look and feel defined by `CELL_SIZE` and `BOARD_SIZE`

- `CELL_SIZE` defines the size of each square cell block.
- `GAME_SPEED` defines the speed of the game (e.g., 8 means the snake advances every 1/8 seconds).
- `BOARD_SIZE_X` and `BOARD_SIZE_Y` define the size of the board in the number of cells for each x and y axis, respectively.
- `LENGTH_PER_FOOD` defines the grow rate of the snake (e.g., 4 means that the length of the snake grows by 4 cells for each food it has eaten).

During the game, we store **the game state** with a set of global variables which represent the current state of the game.

- `snake_head_position_x` and `snake_head_position_y` store the location of the snake's head.
- `snake_direction` stores the direction of the snake. (e.g., 'N' means that the snake is moving toward the north).
- `snake_length` stores the current length of the snake.
- `snake_trail` stores the trail of the snake. (The template source code handles this variable, so you don't need to care about it)
- `food_position_x` and `food_position_y` store the location of the food object.
- `game_over` stores whether the game is over. If the game is over, the game is suspended until the player resets the game.

Your task is to implement the following functions.

Function signature	`create_canvas(nx, ny)`
Parameter 1	`nx (int)`: the number of cells in width
Parameter 2	`ny (int)`: the number of cells in height
Return value	`(cs1graphics.Canvas)` a Canvas object created inside the function 'create_canvas'

This function is used to create a `Canvas` object which is the game board. The width and height of the canvas should be nx × CELL_SIZE and ny × CELL_SIZE, respectively.

Function signature	`initialize_game()`
Return value	`None`

This function is used to initialize the game state variables. It is called when the game is started or reset. The following global variables must be set accordingly:

- `snake_head_position_x` and `snake_head_position_y`: They are set to a random cell index x and y in the game board, where $0 \leq x <$ BOARD_SIZE_X and $0 \leq y <$ BOARD_SIZE_Y.
  ***Hint**: Use `random.randint()` to generate a random number. Refer to this link.
- `snake_direction`: It is set to the closer direction from its head position toward the center of the game board, so that we can prevent accidental game over at the beginning. Refer to Figure 3 for more details.
  ***Note**: You may choose any direction when a conflict occurs while determining "the direction toward the center". For example, if the head position is at the midpoint of the board (when both `BOARD_SIZE_X` and `BOARD_SIZE_Y` are odd numbers), you may choose whatever direction you want.
- `snake_length`: It is set to 1, as the game starts with length 1.
- `snake_trail`: You don't need to manually initialize this variable in this function. Instead, call `initialize_snake_trail()` with newly set `snake_head_position_x` and `snake_head_position_y` as parameters. This function is given in the template source code.
- `food_position_x` and `food_position_y`: You don't need to manually initialize this variable in this function. Call `generate_food()` right after `initialize_snake_trail()` is called. You must implement `generate_food()` as well.
- `game_over`: It is set to False as we're (re)setting the game.

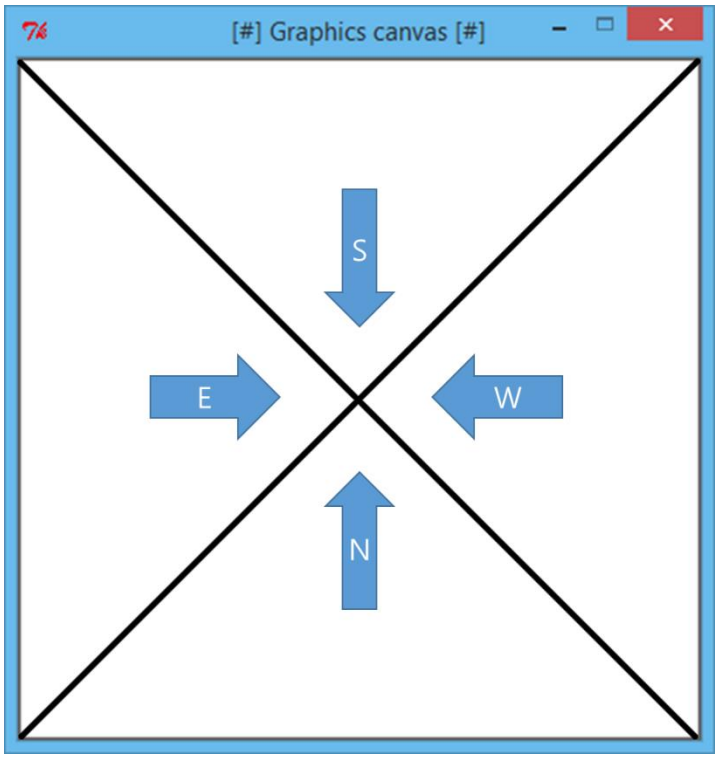

Figure 3 The initial `snake_direction` according to the initial head position

Function signature	`generate_food()`
Return value	`None`

This function is called to create a food object that the snake can eat. We want to avoid the cell where the snake resides, so you should be careful on placing it. Set the global variables food_position_x and food_position_y to the x and y index of the cell you have determined to place it.

***Hint 1**: Use `random.randint()` to generate a random number. Refer to this link.

***Hint 2**: Use the following expression to check if the food object overlaps with the snake:

```
(food_position_x, food_position_y) in snake_trail
```

For example, this loop runs until the food object is out of snake:

```
while (food_position_x, food_position_y) in snake_trail:
    #do something, i.e. randomize the position of food object again
    …
```

Function signature	`update_snake_position()`
Return value	`None`

This function is called for each time step if the game is not over. Proceed `snake_head_position_x` and `snake_head_position_y` by one step ahead in the direction where `snake_direction` indicates. For example, if `snake_direction` is 'N', reduce `snake_head_position_y` by 1.

Function signature	`is_game_over()`
Return value	`(bool) True` if the game over condition is met, `False` otherwise

This function is called to check if the game is over. There are two possible conditions that the game is over:

1) The snake collided to the wall, i.e. One of the variables `snake_head_position_x` and `snake_head_position_y` is out of bound.
2) The snake collided itself.

   Hint: Use the following expression to check if it collides:

   ```
   (snake_head_position_x, snake_head_position_y) in snake_trail[:-1]
   ```

***Hint**: The condition 2 is already implemented in the template code.

Function signature	`create_circle(x, y, color, head = False)`
Parameter 1	`x (int)`: x-axis cell index of the circle
Parameter 2	`y (int)`: y-axis cell index of the circle
Parameter 3	`color (string` or `tuple)`: color of the circle
Parameter 4	`head (bool)`: Indicate if this circle represents the snake's head.
Return value	(`cs1graphics.Circle`) A `Circle` object created inside the function 'create_circle'

Create a circle representing an object on the board with the given color. You should fill the entire cell. For example, if (x, y) is (1, 1), the circle should look like Figure 4. Calculate the center and the radius of `Circle` object accordingly. If `head` is specified as `True`, adjust its depth to smaller value in order that it is still shown, even if it is overlapped with any other circles.

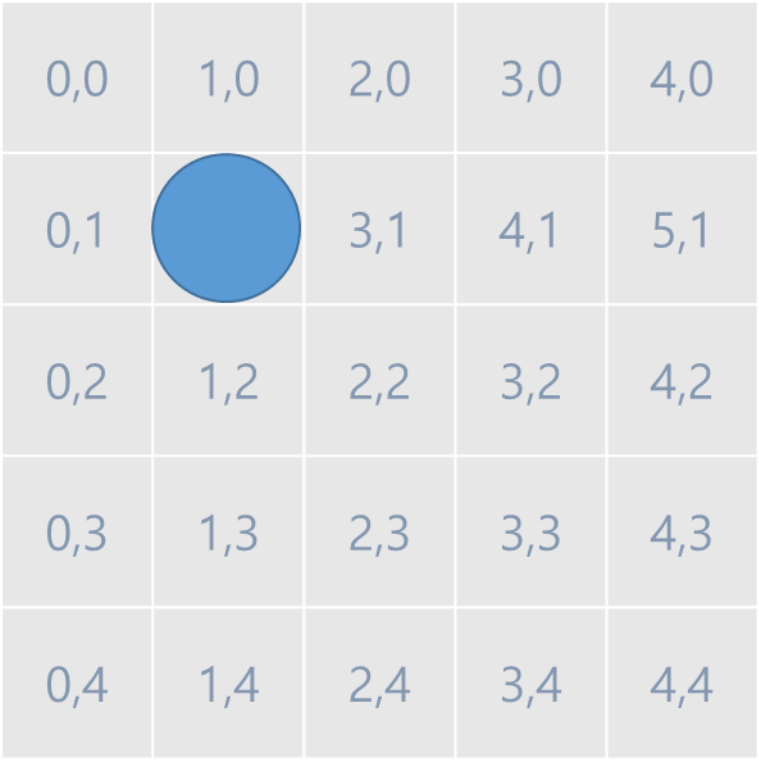

Figure 4 `create_circle`

Function signature	`create_gameover_text()`
Return value	(`cs1graphics.Text`) A `Text` object created inside the function 'create_gameover_text'

Create a text message "GAME OVER" which is printed out at the middle of the canvas. The text should have the smallest depth compared to the other objects in the canvas (i.e. the circles)

***Hint**: Use `help()` to find out how to use `cs1graphics.Text` object.

Function signature	`change_direction(new_direction)`
Parameter 1	`new_direction` (`str`): the direction user intended to change
Return value	`None`

This function is called if the player intends to change the direction. Set `new_direction` to `snake_direction`, if it is a valid turn. It is invalid to change the direction toward the cell which was the snake's last head position (i.e. the snake's "neck"). Refer to Figure 5 for understanding.

*** Hint**: There is `is_this_cell_snakes_neck`() function to check if a cell is the neck of the snake. Pass the next cell to be explored in `new_direction`.

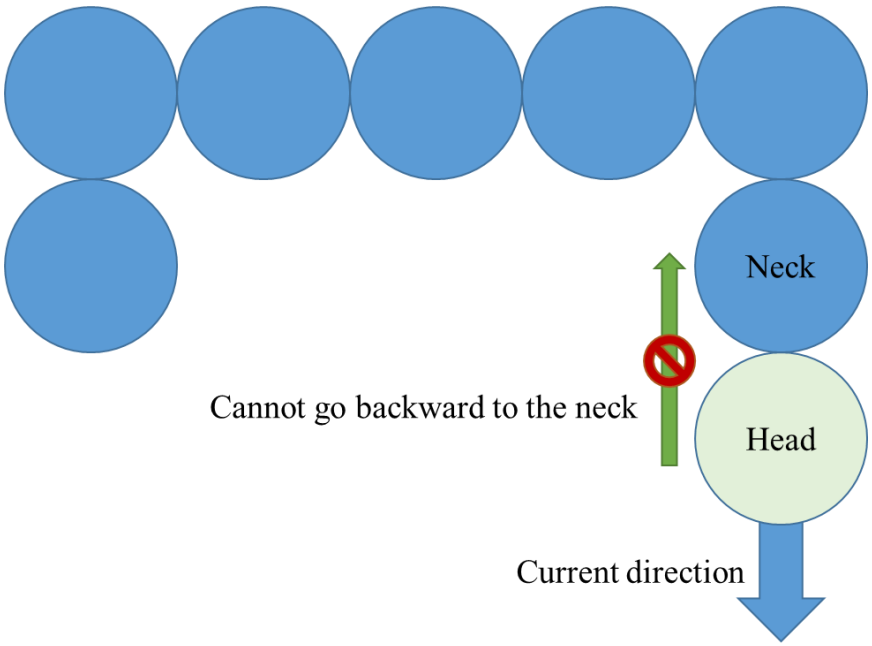

Figure 5 The condition of `change_direction`

CAUTION: DO NOT EDIT ANY OTHER FUNCTIONS, or your score will be reduced. You're allowed to create your own function.

Submission and Evaluation

You need to submit the followings:

- The file 'HW2_your_student_ID.py'
  (e.g.) HW2_20151234.py
- The report 'HW2_your_student_ID.doc', 'HW2_your_student_ID.docx' or 'HW2_your_student_ID.pdf'
  (e.g.) HW2_20151234.doc, HW2_20151234.docx or HW2_20151234.pdf

You MUST archive the source code and the report together into 'HW2_your_student_ID.zip', and then submit the archived file through the webpage for homework submission.

(e.g.) HW2_20151234.zip.

***Note:** You don't need to submit cs101supplement.py.

If you don't follow the submission policy (including but not limited to the file format), you will get penalty.

For the report, there is no strict format, but the report should include the following contents.

- Summary or explanation about the overall algorithm
- Several screen shots of your gameplay
  - The screen shots of the longest snake you can get while you are playing the game is recommended.
  - Choose screen shots which can show processes and results of your program execution, well.
    (e.g.) Initial step, intermediate steps, end step and etc… (2 ~ 4 images are enough.)
- Describe about what you have learnt and what you have felt while doing your homework

The report may contain some parts of your source codes, but please do not include whole source codes.
***Note**: Only *doc*, *docx* and *pdf* files are allowed for your report. Do not submit any *hwp* file.

Your program will be tested in the same environment as Lab computers with a variety of game settings. You will get penalty if your program causes errors, exceptions, malfunction or unsatisfied requirements (e.g., If the snake can turn backward to bite its own neck, your score will be deduced). Delayed homework will get 5% penalty on total score per day from April 13 to April 15. After April 15 23:59 PM, the server will be closed on that day and no more submission will be accepted. Please do it on time.