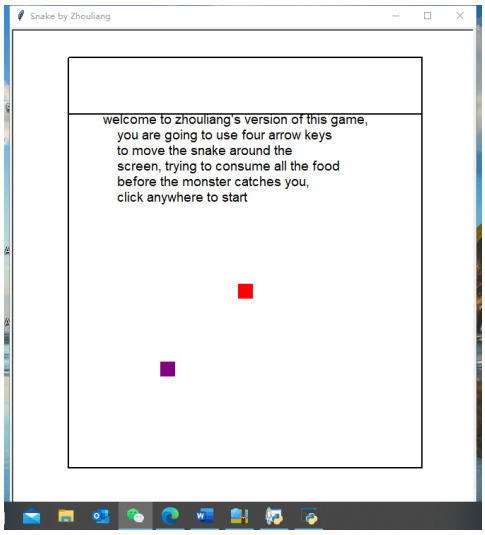
Design Doc

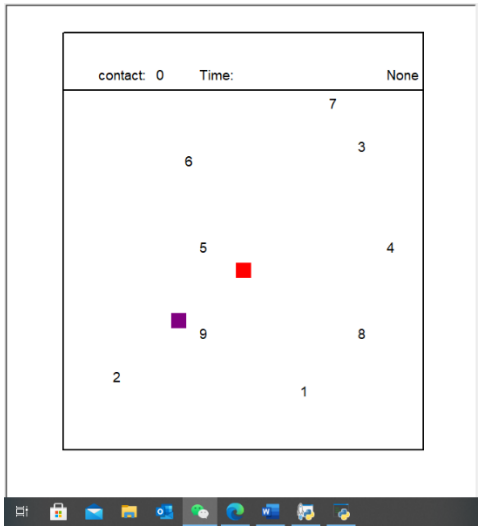
a. Overview:

The game is composed of 3 objects: a snake, a monster and food items represented by a set of numbers from 1 to 9. In the figure shown above, the snake is represented by a sequence of squares where its head and its body are displayed in red and black colors respectively, while the monster by a purple square. The numbers are food items to be consumed by the snake.

The goal of the game is to maneuver the snake within the game area in four directions (up, down, left and right), trying to consume all the food items while avoiding head-on collision with the monster. As each food item is consumed, the snake grows with its body lengthened in size equal to the value of the number being passed. While directing the movement of the snake you should avoid contact with the monster. Furthermore the monster is also programmed to be motioned in the direction towards the head of the snake at a variable speed

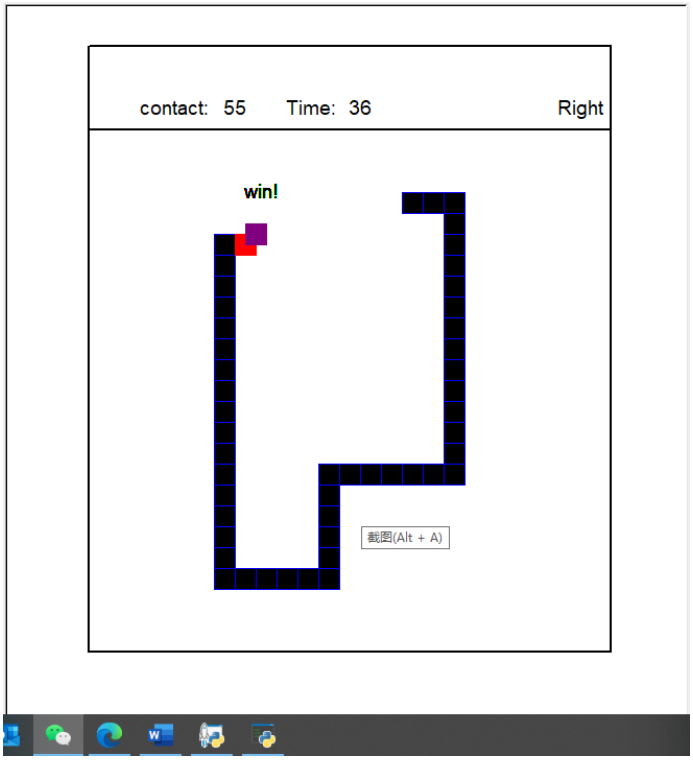
The following screens show the first display of the game (left) and the start of the game (right) after a mouse click:



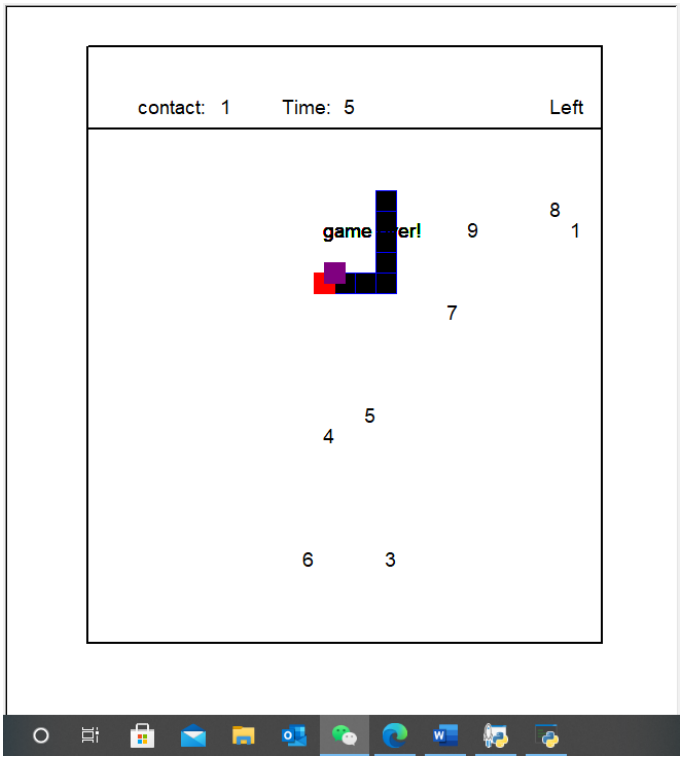


The following 2 screens show the snake with its extended tail after consumption of food items (4), while the monster actively chasing after the snake

The following screen shows the ending of the game after the snake has consumed all the food; shown on the status area are the number of contacts with the monster and the total elapsed game time in seconds:



The following screen shows the ending of the game where the snake collided head-on with the monster and where some food items left unconsumed:



b. Data Model:

Snake and its tail:

1. The ‘g_snake’ is an instance of turtle model, it represents the head of the snake, its shape is square, and with its color red.
2. Initially the position of g_snake is (0, 0). We use the turtle.stamp() method to implement its moving and extension, the size of snake tail is stored in g_snake_size.
3. The heading direction of snake will be represented by each key, key_up, key_down, key_left, key_right means up, down, left, right respectively
4. The position of snake body is stored in g_snakebody_pos and efficient_body_pos as a form of element in list, but the former one is contains all of the position the body of snake have been to at any time, but the latter one is only for one certain moment
5. So each time the snake is going to extend one unit of its length, we simply do ‘g_snake_size += 1’
6. The speed of snake is stored in ‘g_snake_speed’, it is 1/200 at first, each time it consume the food its speed will be slower to 1/210

Food Items:

1. Write the number on the screen: g_food as an instance of turtle.
2. Store the value and position of each food: food_storage as a dictionary, with its key is position of food on the screen and its

value means the how many units the snake is going to grow after swallow this particular food.

c. Program Structure

The snake game is designed using the standard module “turtle”, including the following components:

a. Game Area (status and motion area)

b. A Snake

c. A Monster

d. Food Items

e. Game Status

f. Control

Game Area:

The game area is constructed by the set_screen() function. The game area is composed of an upper area for statuses and a motion area where the snake and monster are moved around, surrounded by a fixed margin along the four sides, with the following dimensions: i. Upper status area = 500 (w) x 80 (h) ii. Lower motion area = 500 (w) x 500 (w) iii. Margins = 80 pixels around

A Snake:

1. To set the head of the snake: set_role()
2. The direction of snake moving is decided by setSnakeheading(key)

and keypressed_dir

3. To move, extend….any operations of snake is conducted in the onTimerSnake() it will be run every 500 ms.
4. To define the appropriate direction: available_move()

A Monster:

1. The monster object is set by set_role()
2. The movement of monster is set by onTimerMonster()

Food Items:

1. The number from 1 to 9, displayed within the motion area in random locations. So each number is initialized in the food_init() function
2. food_display() is to display the remaining food in the food_storage on the screen
3. food_consume() is to implement the result when particular foods are consumed by the snake

Game status:

Motion: motion_display() is to display the ongoing motion of snake on the top right of the status area

Contact:

1. contact_display() is used to display the number of contact of the snake with the monster on the top left of the status area
2. contact_count() is to count the number of contact

Time:

1. time_display() is used to display the running time of the game on the top middle of the status area.

Control:

Use onkey() to bind the key with the directions and pause

d. Processing Logic:

1. The logic used to move the snake and monster:

Snake's moving is operated by the user pressing (up, down, left, right) on their keyboard, when the head of the snake is facing the boundary, it's motion will be stopped, so user should press the appropriate key to get the snake out of this.

The motion of monster is directed by the relative position with the snake, say if the monster is on the upper left of the snake, the program

will compare the distance of snake and monster on both axis, say is they get further on x axis the monster will move left to catch the head of the snake, in the vice case, it will go down, similar for other relative position.

2. The logic to expand the snake tail:

   In my program, I use g_snake_size to memorize the size of the snake, and when the snake is moving, as g_snake is an instance of turtle, I use for loop to implement the growth(using stamp), and the number of loops determines the tails' growth, and the number of loops is the same value as g_snake_size. so say the snake eats one food with certain value, I will increase the g_snake_size by this value.

3. The logic to detect the body contact between snake and the monster:

   While the g_snake stamp, use (Xs, Ys) to implement the position of each body units, and upload this position to the list g_snake_body_pos the g_snake_body_pos[-g_snake_size :] contains the position of each body unit at each time. Compare the position of monster with the element in g_snake_body_pos[-g_snake_size :] if the distance is less than certain value, they contact with each other.

Function Specifications:

1. set_screen: Usage: to set the screen with following requirement:
   i. Upper status area = 500 (w) x 80 (h)

ii. Lower motion area = 500 (w) x 500 (w)

iii. Margins = 80 pixels around

2. set_role(shape=’square’, color=’red’, x=0, y=0)

   this function is used to initialize each turtle object in this program with each of its parameter with default value, shape=’square’, color=’red’, x=0, y=0

3. set_intro()

   write the introduction of this game on the screen

4. food_init()

   initialize the food 1-9 and upload the foods with their value in the dictionary food_storage

5. food_consume()

   when the snake eats food, delete this element in the food_storage

6. food_display()

   display the food in food_storage on the screen

7. time_display()

   before the game finished, display the time elapsed on the status area

8. pause()

   to switch the bool value of g_pause, to show whether the motion of the snake has been paused

9. game_over()

if the monster catches the snake then the game finished. g_defeat = True, means the user lost the game.

Otherwise, g_win = True.

10. available_move()

    when the snake hit the boundary, its motion will be paused, unless the new key input is equal to the motion included in the opp_move list

11. contact_conut()

    count the number of times the snake contact with the monster

12. count_display()

    display the number of contact on the screen

13. setSnakeheading()

    set the heading direction of snake

14. keypressed_dir()

    call the setSnakeheading function

15. onTimerSnake()

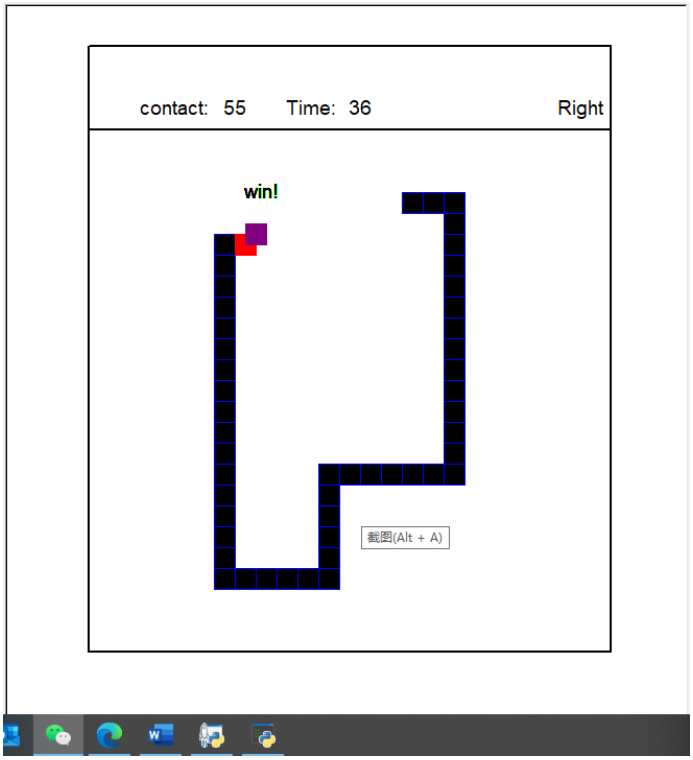
    Call this function every 500 ms. This function is to move the snake with its tail using stamp method, and block its motion when hit the wall. Update the movement of snake on the screen

16. onTimerMonster()

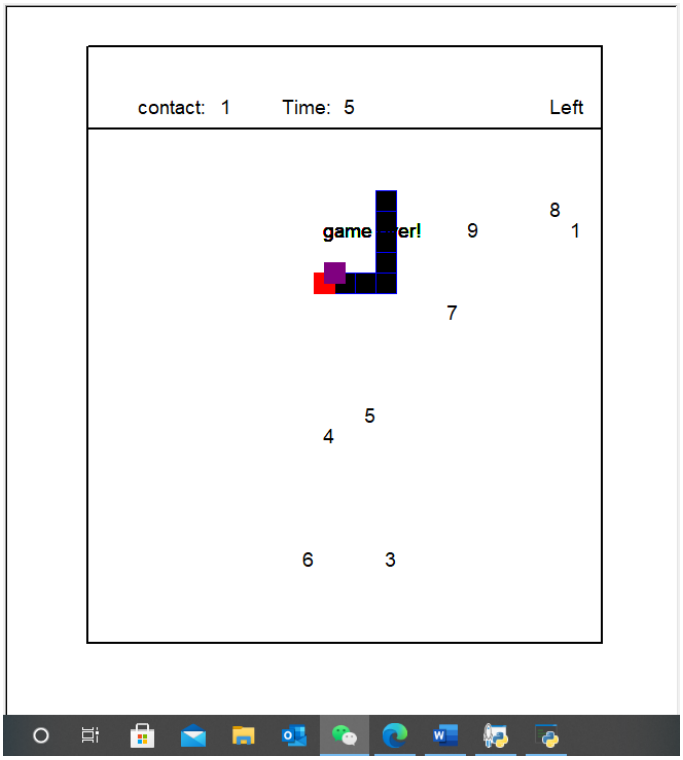
    enable monster to chasing the snake

Output:

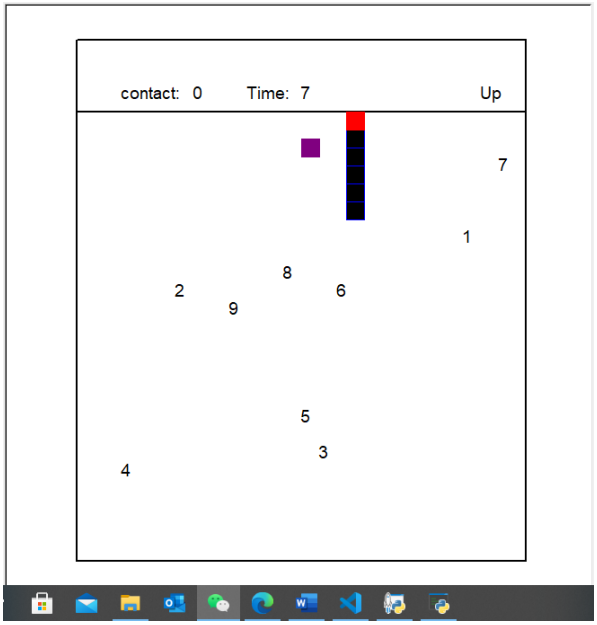
Winner:



Game Over:



With 0 food item consumed:



With 3 food item consumed: