Design documentation of snake game

1. Design

a. Thoughts and overall approach

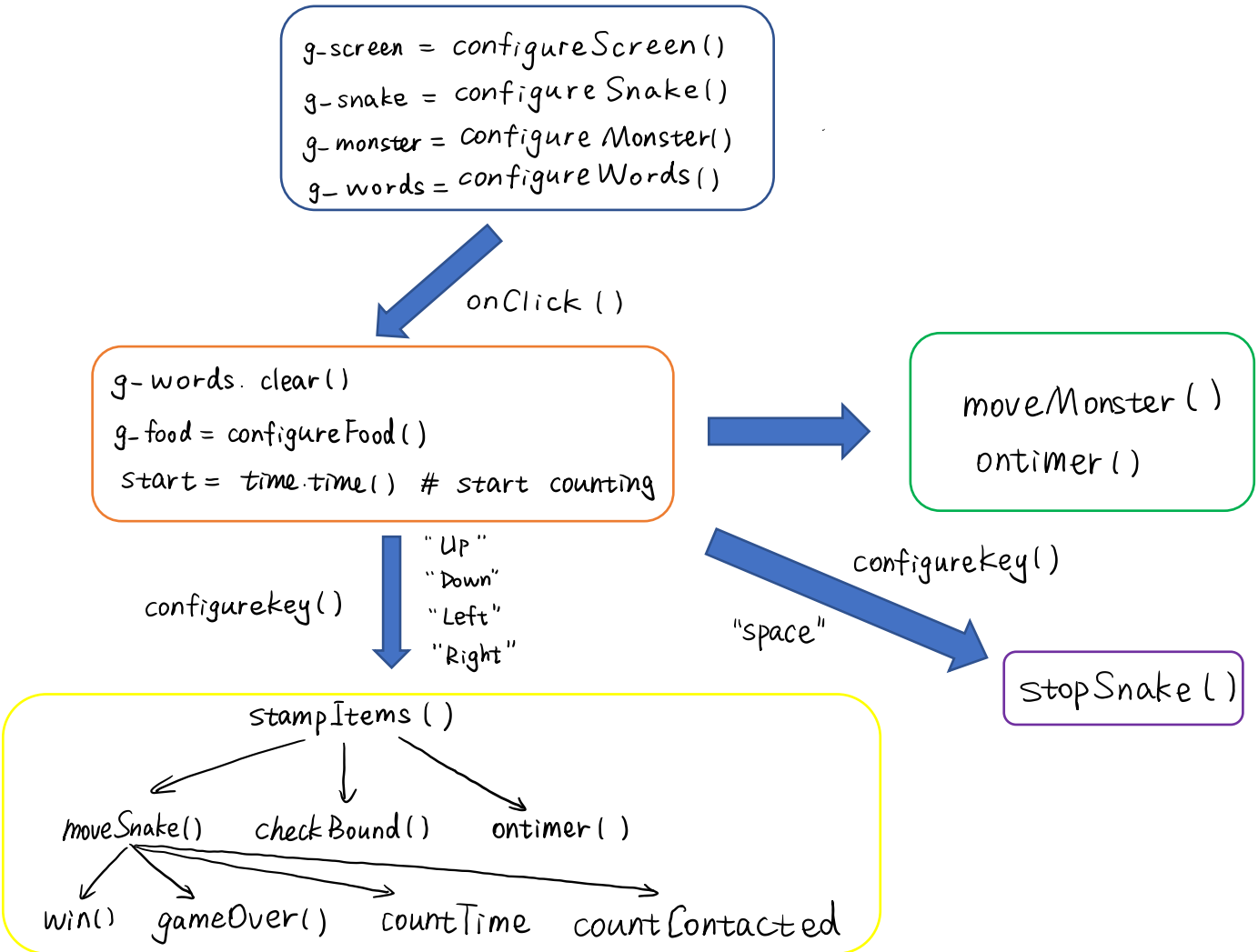
Firstly, I used Turtle() to define snake and monster and food. The whole screen was defined by Screen(). At the beginning of the game, I used a new Turtle() to write the introduction to the game and some instructions for players. The red snake's head was located in the center of the screen. The purple monster was located in a random position which is far away from the snake head.

After the player click a random place on the screen, there will exist nine foods randomly placed on the screen. At the same time, the monster begins to move towards the snake's head. Then the player can control the snake by "Up", "Down", "Left", "Right" on the keyboard. Initially, the length of the snake's tail is five. When the snake's head meet with foods represented by numbers. The number disappears and the length of tails will increase respectively.

If the snake's head touch the boundary of the screen, the whole snake will stop. After the player presses the other directions, the snake will continue to move towards the new direction.

If the snake eats all food, you win the game and the program will inform you that "You Win !!!" If the monster touches the snake's head, you lose the game and the program will inform you that "Game Over !!". After that, the snake and the monster will stop moving, the numbers of time and contacted on title stop too. In addition, the number of contacted means the times when the monster overlaps with the snake's tail, the number of time means the total time you used in seconds.

Structure



b. Data types

Snake

g_snake

The type of it is turtle. I used it to control the head of the snake.

g_lstTail

The type of it is list of turtles. I used it to represent the tail of the snake.

newTail

The type of it is also turtle. I used it to represent the new tail of the snake after the snake eat food. Then the newTail will be added to g_lstTail.

sDirection

The type of it is string. I used it to represent the direction of snake's motion.

init_sDirection

The type of it is also string. I used it to storage the initial direction of snake's motion before it stops. In that case, when the player presses "space" again, the snake could remember the initial direction and keeps moving.

sSpeed

The type of it is integer. I used it to change the speed of the snake after the snake eating food.

Monster

g_monster

The type of it is turtle. I used it to control the monster.

mDirection

The type of it is string. I used it to represent the direction of monster's motion.

mSpeed

The type of it is integer. I used it to change the speed of the monster.

Food items

g_food

The type of it is turtle. I used it to display the position and number of foods on the screen.

g_lstFood

The type of it is list of turtles. I used it to represent the position of foods

newFood

The type of it is also turtle. I used it to represent the foods respectively. The newFood will be added to g_lstFood.

g_lstNumber

The type of it is list of integers. I used it to represent the number of the foods. In addition, g_lstFood and g_lstNumber have one-to-one correspondence.

c. Motion logic

Snake

The motion of snake is controlled by the player. After the player press the button of direction, the snake will move toward the direction at a fixed speed. After the snake eats food, the speed of the snake will decrease slightly. If the snake touches the boundary of the screen, the snake will stop. After the player presses the other directions, the snake will continue to move towards the new direction. When the player presses "space", the whole snake will stop. After the player presses "space" again, the snake will continue to move towards the initial direction.

Monster

The motion of monster is decided by the motion of snake. The monster keeps moving towards the head of the snake. The speed of the monster is slightly faster or slower than the speed of the snake. If the player presses “space”, the motion of monster will not be affected.

d. Expansion logic for the snake tail

When the snake’s head meets with the food, the snake’s tail expands simultaneously. The new tail is the stamp of the previous tail. The tail will move with the head consistently. In addition, when the head of snake stop which means sDirection equals “stop”, the tail also stops.

e. Body contact logic

If the monster overlaps with the snake’s tail, the number of contacted pluses one. If the monster keeps overlapping with the snake’s tail, the number of contacted keeps increasing every time the snake moves.

2. Functions

countTime()

Count the game time in seconds.

countContacted()

Count the number of contacted

configureScreen(w=500, h=500)

Define the whole screen including size and title. w=500 means the width of the screen is 500. h=500 means the height of the screen is 500. Inside the function, I used tracer() to disable auto screen refresh.

configureSnake(shape=’square’, color=’red’, x=0, y=0)

Define the head of the snake including shape, color and position. Inside the function, I used penup() to avoid extra lines. Changing the parameters, it can be used to define monster.

configureMonster()

Define the monster including position, color. Make sure the position of the monster is far enough away from snake.

configureWords(color=’black’, x=-220, y=220)

Define the initial words on screen including color, position, and content. Inside the function, I used hideturtle() to hide the arrow.

configureFood(color=’black’)

Define the food including color, position, and number. The nine foods are placed randomly on the screen. In addition, the length of the snake tail is five. Consequently, I put 5 on (0, 0).

moveUp(), moveDown(), moveLeft(), moveRight()

Change the direction of snake’s motion which means change the content of sDirection.

moveMonster(d=20)

Move the monster through ontimer(). d=20 means the distance of one move is 20. The speed of the monster is slightly faster or slower than the speed of snake. x means the difference between snake’s head and monster in x axis. y means the difference between snake’s head and monster in y axis. Compare the absolute value of x and y, it will lead to different direction.

moveSnake(d=20)

Move the snake and refresh the title and check the game status. d=20 means the distance of one move is 20

gameOver(color=’red’)

Check the status of the game through the position connection between snake's head and monster. If they overlap with each other, write "Game Over !!" and stop the motion of all items. color='red' means the color of "Game Over !!" is red.

stopSnake()

Change the direction of snake's motion to stop or restart the motion of snake.

win(color='red')

Check the status of the game through the length of the snake's tail. If the length of tail equals fifty, write "You Win !!!" and stop the motion of all items. color='red' means the color of "You Win !!!" is red.

configureKey(s)

Bind moveUp(), moveDown(), moveLeft(), moveRight() with four arrow keys and "Space Bar". s is the screen.

stampItems()

Add the tail to the snake's head and enable the tail to move with the head consistently. After the snake eat food, the speed of the snake will decrease.

checkBound()

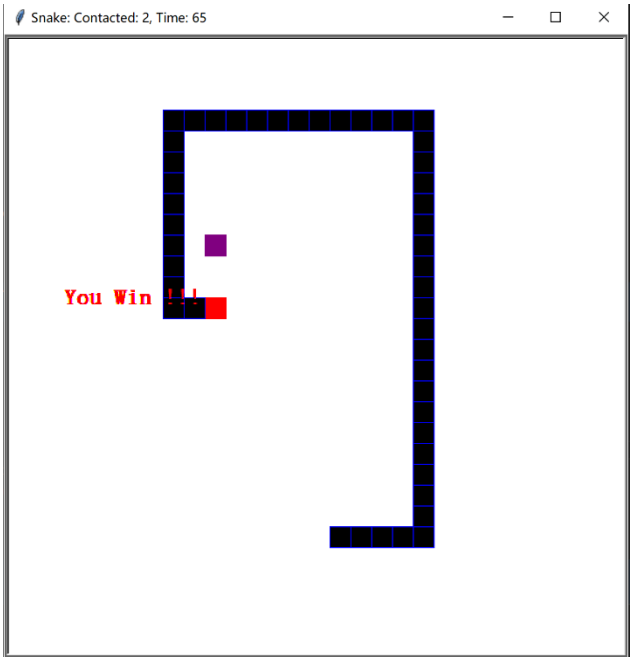
Check whether the snake touches the boundaries of the screen or not. If yes, stop the whole snake.

onClick(x,y)

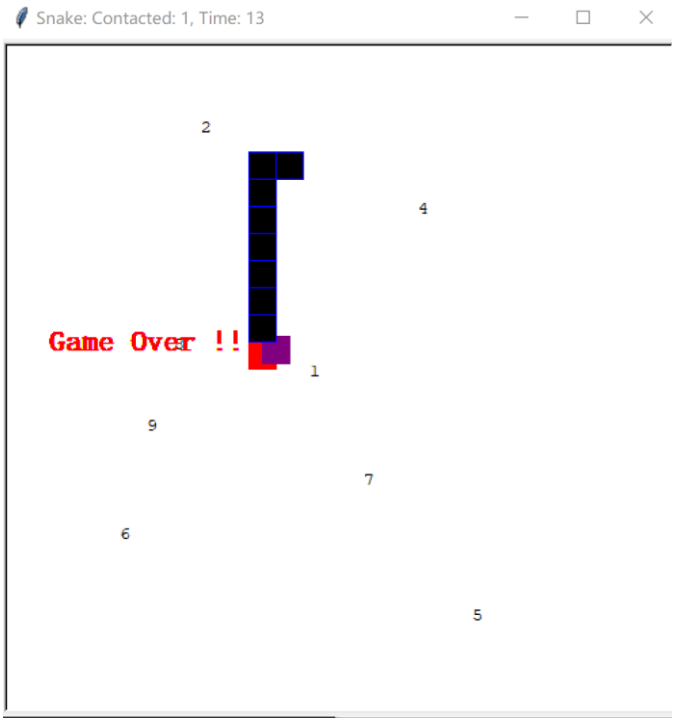
After the player click the screen, the words are cleared and the foods exist. Monster begins to move.

3. Output

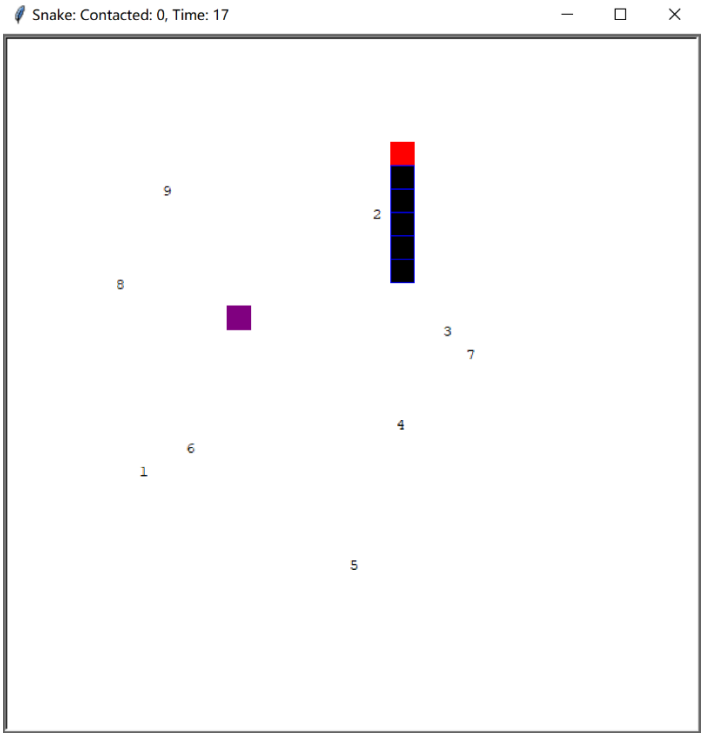
Winner



Game over



With 0 food item consumed



With 3 food items consumed