CSC1002 Computational Laboratory

Snake Game – 2021
Design Document

Name: Dai Yulong

Student ID: 119010054

The Chinese University of Hong Kong, Shenzhen

Overview

In this assignment, I have designed and developed a Snake game. The whole game is composed of 3 objects: a snake, a monster and food items represented by a set of numbers from 1 to 9. Firstly, we need to create a 660 * 740 screen. In this screen, there is a 500 * 500 margin area, which displays the game. Above the motion area is a status area. It will show the number of contacts, count of time, and the real-time motion status. After clicking, the introduction is hidden and the game starts. The snake begins to extend (the initial length is 6). At the same time, the monster begins to chase the snake. The snake moves in the direction according to the instruction given by the user. The player controls the snake to eat more and more food as they can. Once all foods are eaten the player will win the game. And if the monster catches up with the snake, the player loses the game.

Data Model

Here are some important data types.

<1> **g_screen**: turtle.screen() type

It's used to draw the screen, relate the movements with the keyboard, update the screen and install timers for functions

<2> **g_snake**: turtle type

It's the head of the snake. And it controls the snake heading direction.

<3> **g_num**: turtle type

It controls the display of both introduction and number.

<4> **g_tip**: turtle type

It displays the real-time status context, which includes contacts, time, and real-time motion.

Program Structure

In the beginning, the program will call the "configure" functions to draw the screen. After clicking on the screen, the program call the main() function to execute the game. Every time player taps Up, Down, Left, Right or Blank Space key, the program will call the corresponding up(), down(), left(), right(), and pause(). Then it calls moveUp(), moveDown(), moveRight(), moveLeft(). And these functions call moveSnake() function to move the snake. When moving the snake, it calls judge() function to detect the foods in front of the snake head. At the same time, monsterMove() makes the monster chase the snake. And the contact function counts the number of contacts between snake and monster. checkOver() function checks whether the game is over.

Processing Logic

<1> Snake Motion

Its motion depends on the operation of the user. The user taps Up, Down, Left, Right or Blank Space key to set the snake into the motion they want.

<2> Monster Motion

Get the position of the snake and the monster and move in the direction of the x-axis when $|x1-x2|>|y1-y2|$ and move in the direction of the y axis otherwise. And whether to move in the negative or positive direction of the x-axis depends on whether $x1>x2$ or $x1<=x2$.

<3> Snake Tail Extension

If the snake passes an area with no food within, its last part (of length 1) of the tail will be removed and the head will move forward. If the snake passes an area with food within, its last part of the tail will not be removed until the snake moved the same steps of that number.

<4> **Body contact**

A list is created to record the position of each part of the snake. And then we check whether each part of the snake shares contacts with the monster. If it is the tail, keep on moving and add the contact numbers. If it is the head, the game is over.

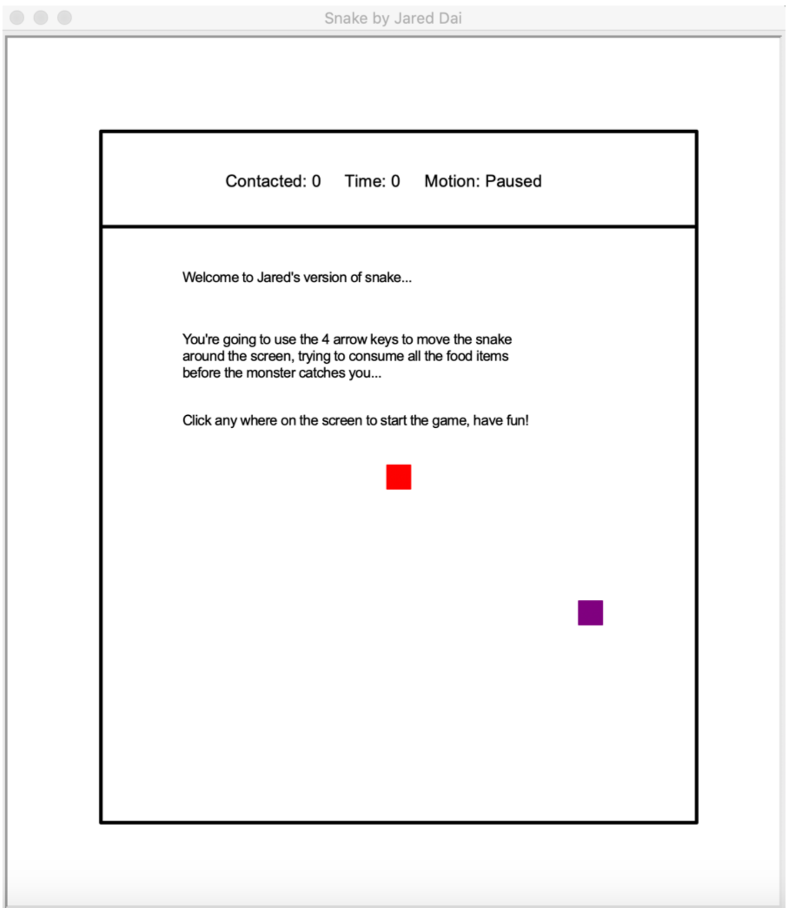
Functional Spec

1. configureMargin(): draw the margin area
2. configureScreen(): create the screen of width 660 and height 740
3. configureSnake(): draw the head of the snake
4. configureMonster(): draw the monster
5. configureText(): draw the introduction part
6. generateFood(): the list all_pos[] stores all coordinates of motion area. In the beginning, every value in the list is initialized to 0. And after randomly generating the foods, we will get 9 coordinates with number 1~9. Then update the corresponding value in the list all_pos[]
7. pause(): pause the motion of the snake or set the snake into moving again.
8. contact(x, y): check whether the monster contacts with the snake and count the number. (x, y) is the position of the monster. Global variable coor_snakeBody stores all coordinates of body squares of the snake. We can determine whether there exists a contact by calculate the distance between them.
9. judge(x,y): judge whether there is a food (or more) in this area. (x,y) is the position of snake head. It detects the area of 30 * 30 around the snake head.
10. moveSnake(): determine the different status and move the snake.
11. moveUp() / moveDown() / moveLeft() / moveRight(): move the snakes head in the specified direction
12. monsterMove(): move the monster automatically. And the status area is updated here.

13. up() / down() / left() / right(): monitor the status of snake. If the status is paused, pressing any of the four arrow keys will un-pause the snake motion and move in the direction of the arrow key being pressed. Global variables pre_move1 / pre_move2 / pre_move3 / pre_move4 store the previous moving condition. For example, if the last status of snake is moving up, pre_move1 is set to true and the rest are false.
14. checkOver(): check whether the game is over.
15. configureKey(): relate the movement with the keyboard
16. main(): include lots of things we need to do after clicking the screen.

Sample Output

The Introduction Screen



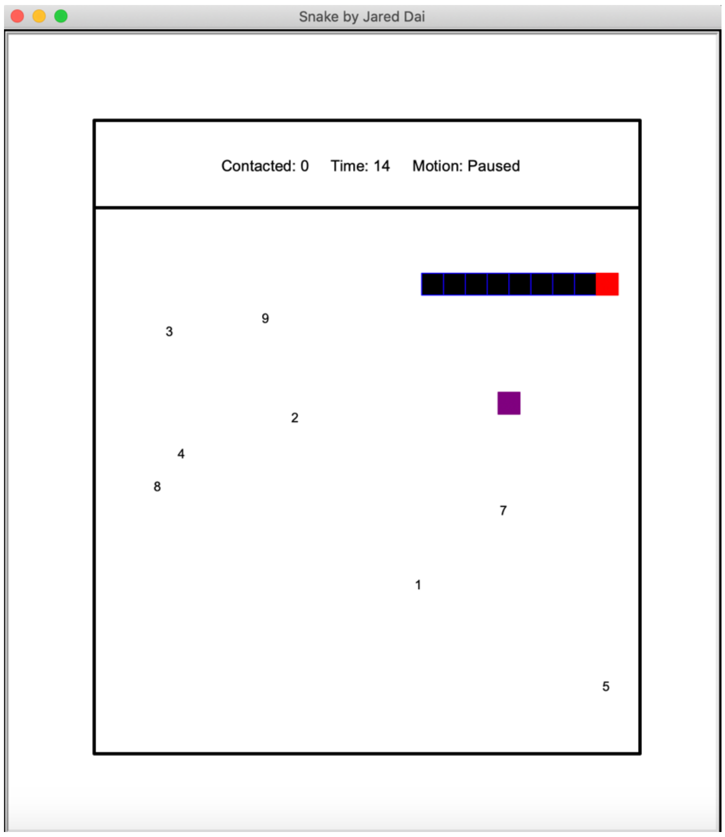
Contacted: 0 Time: 0 Motion: Paused

Welcome to Jared's version of snake...

You're going to use the 4 arrow keys to move the snake
around the screen, trying to consume all the food items
before the monster catches you...

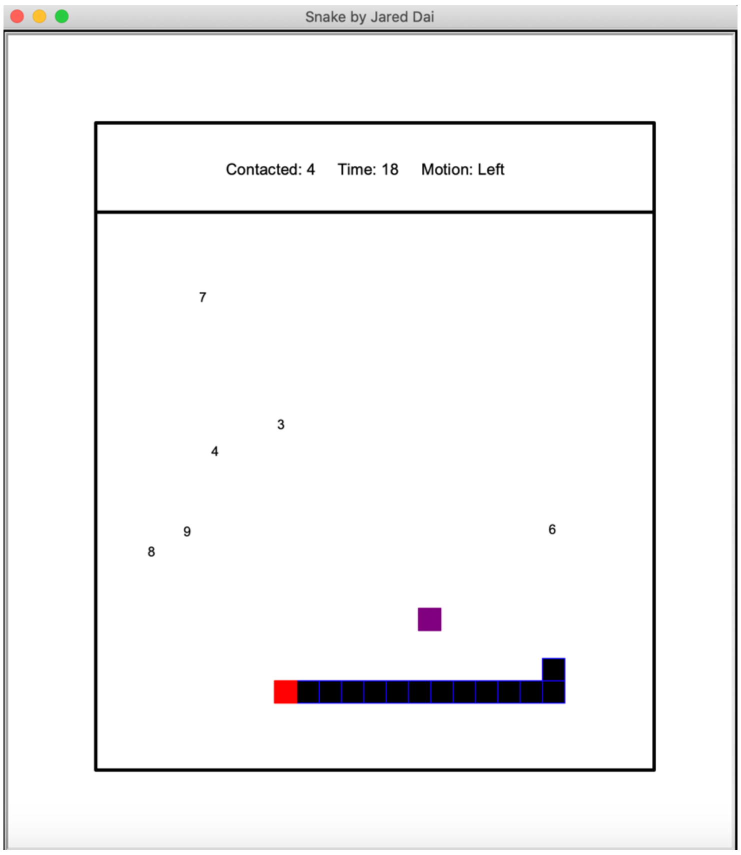
Click any where on the screen to start the game, have fun!

Snake Paused



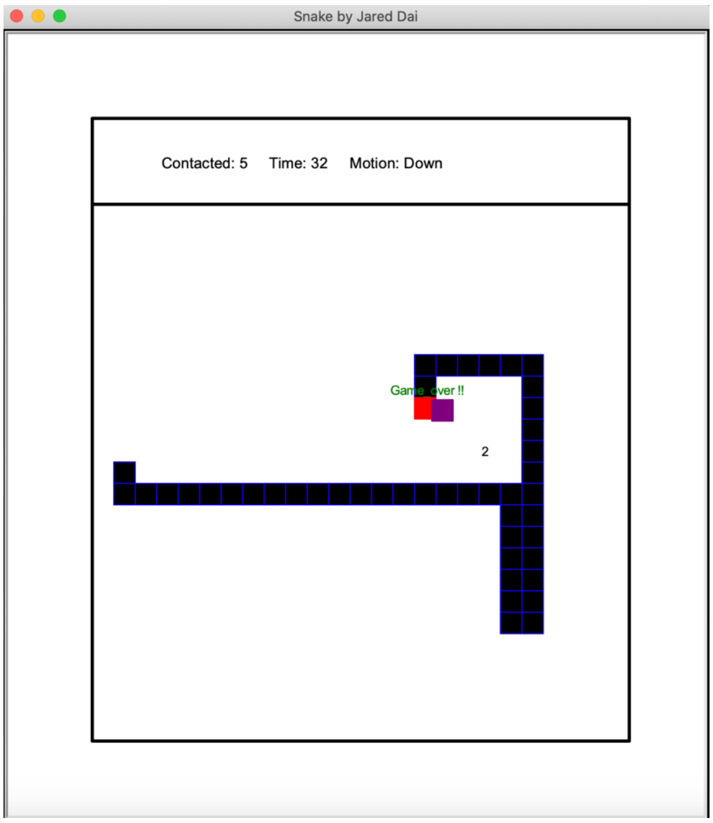
Contacted: 0 Time: 14 Motion: Paused

Three Foods Consumed



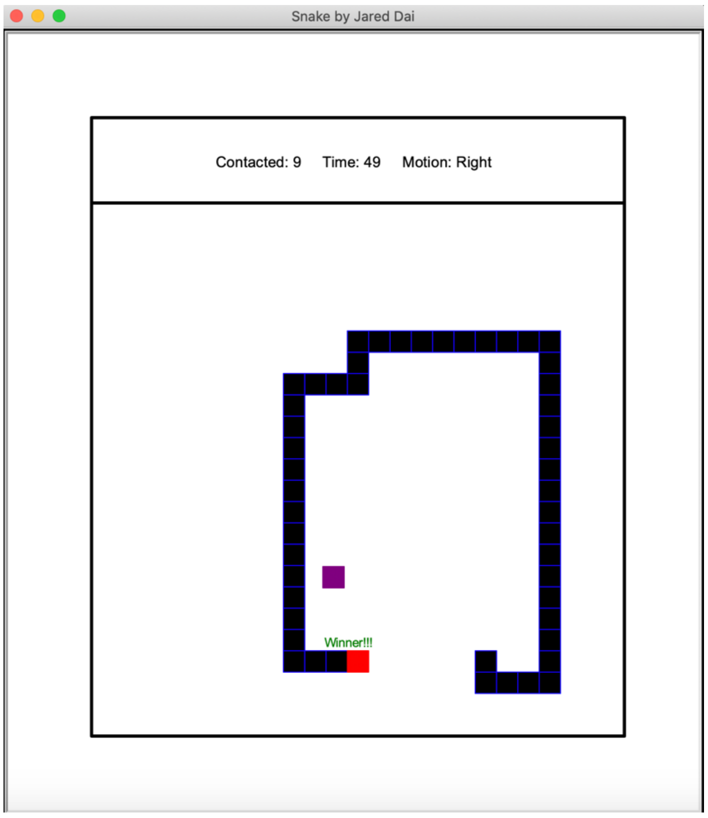
Contacted: 4 Time: 18 Motion: Left

Game Over !!



Contacted: 5 Time: 32 Motion: Down

Winner !!!



Contacted: 9 Time: 49 Motion: Right