# DESIGN DOCUMENTATION

## 1.Design

### 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.

### b. Data Model

### i. turtle

#### 1. Snake's head

Turtle()

#### 2. Monster

Turtle()

#### 3. Screen

Screen()

#### 4. Others

I use lots of Turtle() to draw frames, words, etc.

### ii. Others

**1. Food items**

list() for coordinates and a Turtle() for each food.

**2. Body information**

list() for coordinates of snake body, easy to update

**3 Others**

Many int, bool, str for flags, words, and loop control

## c. Program Structure

i. Configure screen
ii. Configure snake, monster, and food
iii. Start, move, update screen and judge (win or lose).

## d. Processing Logic

### i. Motion of snake

In unit time(of snake), snake will forward 20 pixels. And user can use key board to change its orientation.

### ii. Motion of monster

Program will compare the gap of x and y direction and move the monster to the snake.

### iii. Expand the tail

The program will generate 9 random coordinates and set them in a list. Once they were eaten by snake, they will be deleted.

### iv. Contact between snake and monster

Every time monster moved, the screen will judge whether they are contact, that is, check whether the difference of their coordinates are less than 20. The cordinates of body can be got from g_bodyInfor

# Function Specifications

## Configure Part

```
start(x,y)
```

Start the game. Will be executed after the user click the screen.

Also, it  will **start counting time**, clear the introduction message, generate food items, generate the Turtle() which stamp the body of snake, start move and make click on the screen useless(when click screen, nothing more will happen).

```
configure_screen()
```

Generate the standard screen, return the Turtle()

```
configure_snake()
configure_monster()
```

Generate the head of snake and monster, return the Turtle()

```
configure_intro()
```

Write the information of game, return the Turtle().

```
configure_boundary()
```

Generate boundary

```
configure_food()
```

Generate 9 food at 9 random places and draw them in corresponding places. Return a list of food coordinates(tuple) and a list of Turtle() which draw the food. And pay attention to avoid overlapping.

```
body_writer()
```

Generate a Turtle() for stamp the body of snake

```
generate_bodyInfor()
```

Append and delete body information to keep body move in appropriate

## Move Part

```
snake_move()
```

Move snake, both head and tail, and update screen every unit time. Unit time is determined by SNAKE_SPEED. If the length of body is changing, the speed will be slower.

```
body_move()
```

If the game is on, move the body.

```
eat()
```

Check whether snake eat the food. Cancel the food eaten.

```
print_body()
```

As it was called.

```
head_move()
```

Check whether the snake will touch barriers. Forward 20 pixels every unit time if game is on. Detect whether the user press the key.

```
up()
down()
left()
right()
pause()
```

As they are called. They will change the list of input. The last move will be implenmented.

### Judgment Part

```
win()
lose()
```

Use length of body to determine win, the distance between snake and monster to determine lose.

```
update_game_status()
```

Check win or lose. Update information.

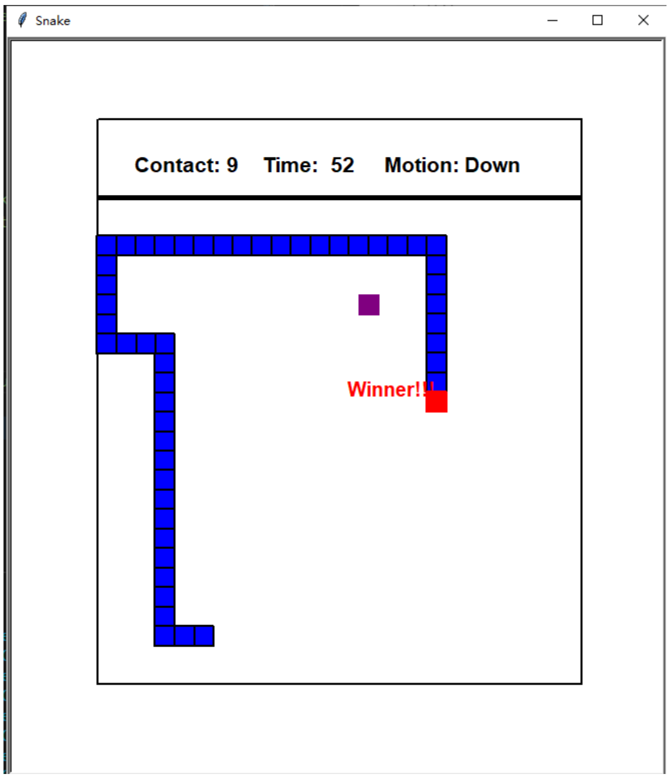
```
check_contact()
```

Check whether they contact. If so, contact will add 1.

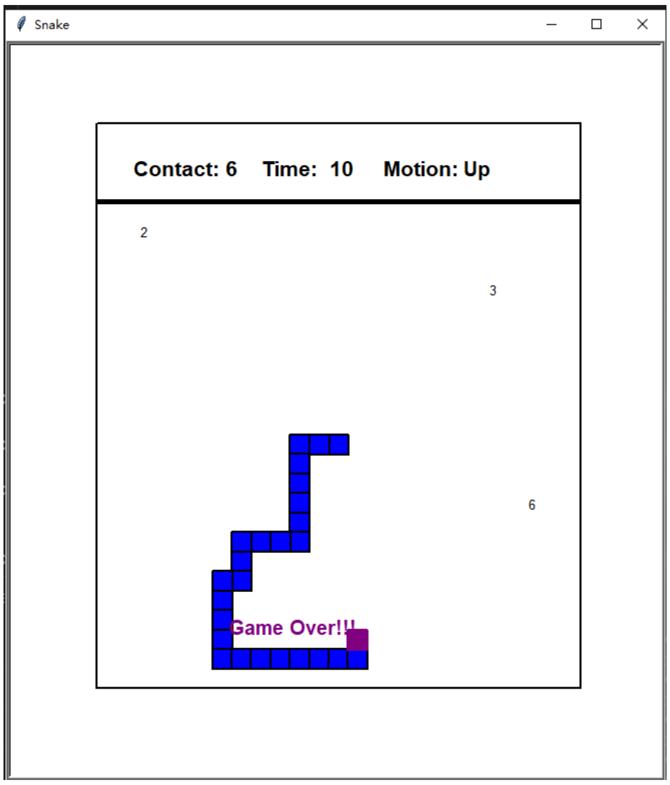
## 3.Output

### a.

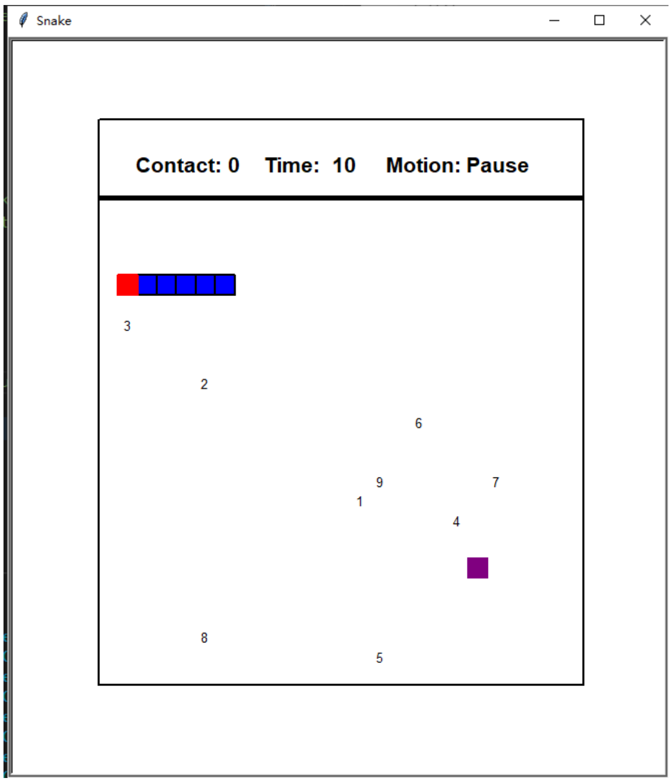
## i. Winner

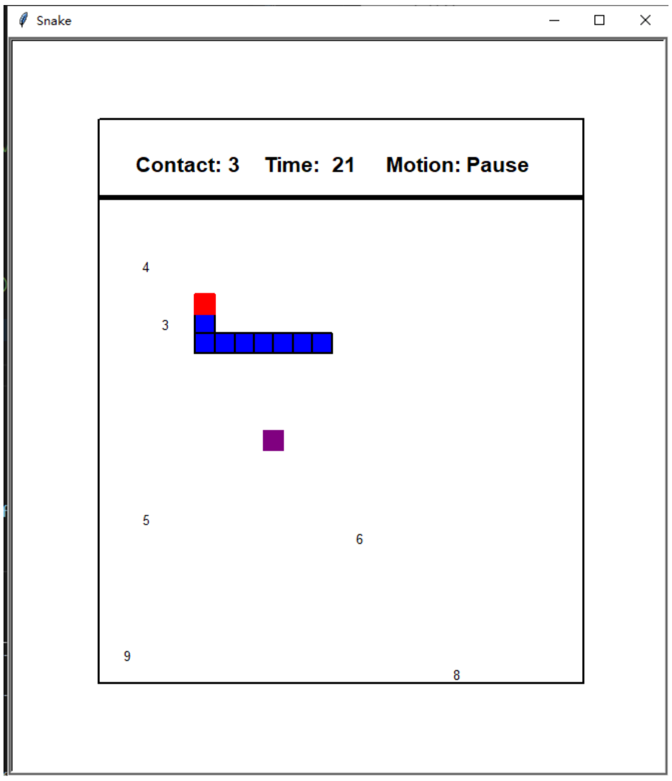


## ii. Game over



## iii. 2 others showing various stages of the game





## 4. Some Details

The Usage of some global variables are stated in comments

The speed of monster was random around the snake.