**Design Doc**

**OVERVIEW**

The program is saved in A2\_SSE\_120090246\_Source.py, I designed and developed an interactive game that 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 are 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. When all foods are eaten, then you win the game.

**DATA MODEL**

Boolean type

List

Dictionary

Tuple

**PROGRAM STRUCTURE**

Import Turtle and Screen from turtle.py.

Set initial state of the board and the snake and the timer.

Generate the screen and boundary from turtle.

Initialize a snake and a monster.

Define the method for the snake to eat and the monster to move.

Check whether the snake has eaten all foods while the monster has not caught the head of the snake.

**PROCESSING LOGIC**

First, generate a turtle graphic screen with a title. Then draw the segments on the screen using “.goto” function to form a rectangular space. And then I print the welcome text and the box on the top of the board.

Second, I gives the commands for the computer to react when I press the keyboard. I use “.onkey” function to execute the corresponding function when I press the key on the keyboard. It is moving upwards, downwards, leftwards, and rightwards. Besides, I define a snake and a monster on the screen. The snake appears in random direction and the monster appears at random position. What’s more, I generate number from 1 to 9 at random positions with x and y coordinates generated randomly for the snake to eat. If the snake pass through it, the number will disappear, because the corresponding element will be removed from the list.

When all of the numbers are eaten, you win the game. If the monster catches the head of the snake, the you lose the game. You can also press space key to pause or continue the snake’s movement.

**FUNCTIONAL SPEC**

**Part 1**

1. generate\_screen():

print the full screen

1. generate\_region():

use the pen to draw segments on the screen

1. generate\_turtle():

Give the common formula for every objects to be shown.

Input parameter: show: boolen, whether to be shown

Color: string, color to be used

Pos: tuple, position to be put.

Output: the turtle graphics.

1. generate\_countings():

set the “contact”, “time”, and “motion” of the snake

1. welcome\_text():

Set the welcome text in move area.

Output: the tests on the top.

**Part 2**

1. GO\_UP():

GO\_DOWN():

GO\_LEFT():

GO\_RIGHT():

These four functions gives the move direction.

1. PRESS\_KEY():

The function reacts to the key pressed on the keyboard, and executive the order, respectively.

1. set\_food():

The function can put numbers on the screen randomly for the snake to eat.

Input: None

Output: numbers from 1 to 9 on the screen.

1. get\_pos\_xy():

Get x and y coordinates of the turtle graphic.

1. food\_remain():

if there is no food left, then return True

1. eat\_food():

Realize the function of eating food.

Parameters: numbers: number to be eaten

Pos of that number

1. set\_timer():

count the total time and change every second.

1. Screen\_update():

Update the screen every 100ms.

1. A\_snake():

Define a global snake to move.

1. A\_monster():

Define a global monster to move.

1. Snake\_move():

Give a formula for the snake to move forward and eat the food.

If the game state is “run”, then, the snake will move forward.

1. monster\_move():

give a formula for the monster to move towards the head of the snake.

**Part 3**

1. start\_Game():

initialize the game status and click on the screen to start

two parameters: x and y

1. PAUSE():

Click “space” to pause or continue the game.

1. end\_Game():

check whether you win the game and print the result of the game.

**SAMPLE OUTPUT**

**As shown in the next page.**





