**Tae Hoon Kim**
**20161993**
Johnseen16@gmail.com

# Scary Snake Game Review and Explanation

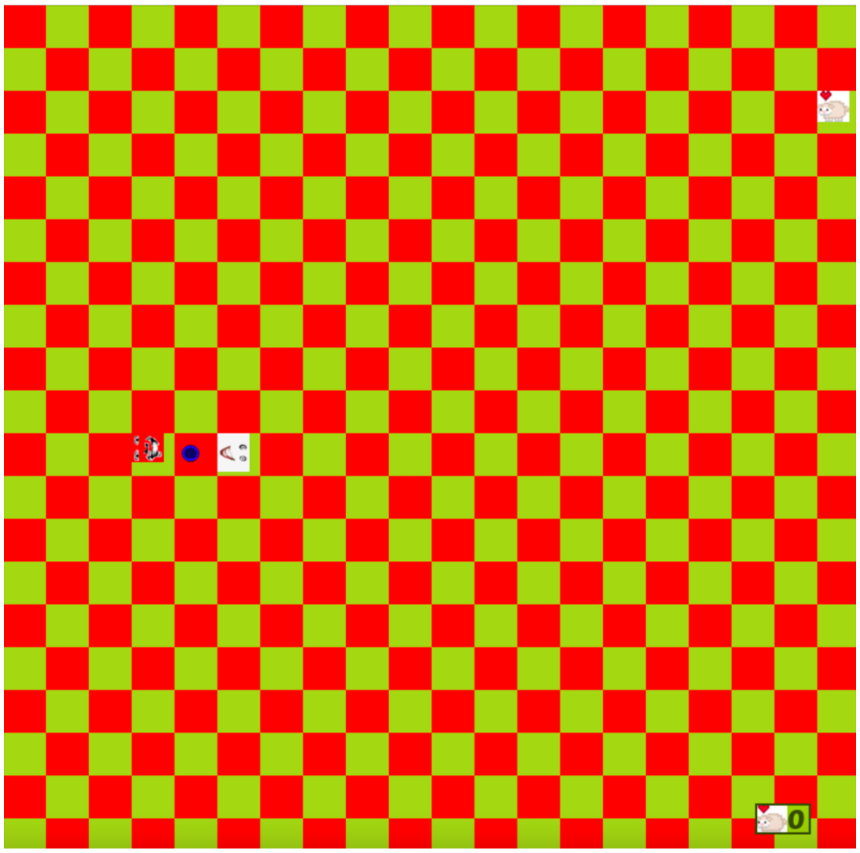
```
import pygame
import os
import sys
import random
from pygame.math import Vector2
from time import sleep

class SNAKE:
    def __init__(self):
        self.body = [Vector2(1,10),Vector2(2,10),Vector2(3,10)]
        self.direction = Vector2(0,0)
        self.new_block = False

        #convert() or convert_alpha() and they will improve game performance, as they convert the image to draw it faster.
        #So, transparent iamge will be black and white. Making it blend with the background.
        self.head_up = pygame.image.load('/Users/taehoonkim/Downloads/3_Pygame/assets/head_up.png').convert_alpha()
        self.head_down = pygame.image.load('/Users/taehoonkim/Downloads/3_Pygame/assets/head_down.png').convert_alpha()
        self.head_right = pygame.image.load('/Users/taehoonkim/Downloads/3_Pygame/assets/head_right.png').convert_alpha()
        self.head_left = pygame.image.load('/Users/taehoonkim/Downloads/3_Pygame/assets/head_left.png').convert_alpha()

        self.tail_up = pygame.image.load('/Users/taehoonkim/Downloads/3_Pygame/assets/tail_up.png').convert_alpha()
        self.tail_down = pygame.image.load('/Users/taehoonkim/Downloads/3_Pygame/assets/tail_down.png').convert_alpha()
        self.tail_right = pygame.image.load('/Users/taehoonkim/Downloads/3_Pygame/assets/tail_right.png').convert_alpha()
        self.tail_left = pygame.image.load('/Users/taehoonkim/Downloads/3_Pygame/assets/tail_left.png').convert_alpha()

        self.body_vertical = pygame.image.load('/Users/taehoonkim/Downloads/3_Pygame/assets/body_vertical.png').convert_alpha()
        self.body_horizontal = pygame.image.load('/Users/taehoonkim/Downloads/3_Pygame/assets/body_horizontal.png').convert_alpha()

        #the body was suppose to turn by the different body part. But its round..
        self.body_tr =
pygame.image.load('/Users/taehoonkim/Downloads/3_Pygame/assets/body_tr.png').convert_alpha()
        self.body_tl =
pygame.image.load('/Users/taehoonkim/Downloads/3_Pygame/assets/body_tl.png').convert_alpha()
        self.body_br =
pygame.image.load('/Users/taehoonkim/Downloads/3_Pygame/assets/body_br.png').convert_alpha()
        self.body_bl =
pygame.image.load('/Users/taehoonkim/Downloads/3_Pygame/assets/body_bl.png').convert_alpha()
        self.crunch_sound = pygame.mixer.Sound('/Users/taehoonkim/Downloads/3_Pygame/assets/bgm.wav')

    def draw_snake(self):
        #it is used to draw the snake.
        self.update_head_graphics()
        self.update_tail_graphics()

        #the position of the snake.
        for index,block in enumerate(self.body):
            x_pos = int(block.x * cell_size)
            y_pos = int(block.y * cell_size)
            block_rect = pygame.Rect(x_pos,y_pos,cell_size,cell_size)

            if index == 0:
                screen.blit(self.head,block_rect)
            elif index == len(self.body) - 1:
                screen.blit(self.tail,block_rect)
            else:
                previous_block = self.body[index + 1] - block
                next_block = self.body[index - 1] - block
                if previous_block.x == next_block.x:
                    screen.blit(self.body_vertical,block_rect)
                elif previous_block.y == next_block.y:
                    screen.blit(self.body_horizontal,block_rect)
                else:
                    if previous_block.x == -1 and next_block.y == -1 or previous_block.y == -1 and next_block.x == -1:
                        screen.blit(self.body_tl,block_rect)
                    elif previous_block.x == -1 and next_block.y == 1 or previous_block.y == 1 and next_block.x == -1:
                        screen.blit(self.body_bl,block_rect)
                    elif previous_block.x == 1 and next_block.y == -1 or previous_block.y == -1 and next_block.x == 1:
                        screen.blit(self.body_tr,block_rect)
                    elif previous_block.x == 1 and next_block.y == 1 or previous_block.y == 1 and next_block.x == 1:
                        screen.blit(self.body_br,block_rect)

    def update_head_graphics(self):
        head_relation = self.body[1] - self.body[0]
        if head_relation == Vector2(1,0): self.head = self.head_left
        elif head_relation == Vector2(-1,0): self.head = self.head_right
        elif head_relation == Vector2(0,1): self.head = self.head_up
        elif head_relation == Vector2(0,-1): self.head = self.head_down
```

```python
    def update_tail_graphics(self):
        tail_relation = self.body[-2] - self.body[-1]
        if tail_relation == Vector2(1,0): self.tail = self.tail_left
        elif tail_relation == Vector2(-1,0): self.tail = self.tail_right
        elif tail_relation == Vector2(0,1): self.tail = self.tail_up
        elif tail_relation == Vector2(0,-1): self.tail = self.tail_down

    def move_snake(self):
        if self.new_block == True:
            #it copies the body to enlarge the snake.
            body_copy = self.body[:]
            body_copy.insert(0,body_copy[0] + self.direction)
            self.body = body_copy[:]
            self.new_block = False
        else:
            body_copy = self.body[:-1]
            body_copy.insert(0,body_copy[0] + self.direction)
            self.body = body_copy[:]

    def add_block(self):
        self.new_block = True

    def play_crunch_sound(self):
        self.crunch_sound.play()

    def reset(self):
        # it will go back to the specified vector.
        self.body = [Vector2(5,10),Vector2(4,10),Vector2(3,10)]
        self.direction = Vector2(0,0)


class MEAT:
        #randomly placed.
    def __init__(self):
        self.randomize()

    def draw_meat(self):
        meat_rect = pygame.Rect(int(self.pos.x * cell_size),int(self.pos.y * cell_size),cell_size,cell_size)
        screen.blit(sheep,meat_rect)
        #pygame.draw.rect(screen,(126,166,114),meat_rect)

    def randomize(self):
        #the meat is moved randomly by -1 cell
        self.x = random.randint(0,cell_number - 1)
        self.y = random.randint(0,cell_number - 1)
        self.pos = Vector2(self.x,self.y)
```

```
class MAIN:
    def __init__(self):
        self.snake = SNAKE()
        self.meat = MEAT()

    def update(self):
            #three important parts, the snake, collision of the meat, and when it hits the out side or eat itself.
        self.snake.move_snake()
        self.check_collision()
        self.check_fail()

    def draw_elements(self):
        #drawing the elements of character
        self.draw_hell()
        self.meat.draw_meat()
        self.snake.draw_snake()
        self.draw_score()

    def check_collision(self):
        #check_collision allows the snake to detect and erase the meat.
        if self.meat.pos == self.snake.body[0]:
            self.meat.randomize()
            self.snake.add_block()
            self.snake.play_crunch_sound()

        for block in self.snake.body[1:]:
            if block == self.meat.pos:
                self.meat.randomize()

    def check_fail(self):
        if not 0 <= self.snake.body[0].x < cell_number or not 0 <= self.snake.body[0].y < cell_number:
            self.game_over()

        for block in self.snake.body[1:]:
            if block == self.snake.body[0]:
                self.game_over()

    def game_over(self):
        self.snake.reset()

    def draw_hell(self):
        hell_color = (241,21,20)
        for row in range(cell_number):
            if row % 2 == 0:
                for col in range(cell_number):
                    if col % 2 == 0:
                        hell_rect = pygame.Rect(col * cell_size,row * cell_size,cell_size,cell_size)
                        pygame.draw.rect(screen,hell_color,hell_rect)
```

```python
            else:
                for col in range(cell_number):
                    if col % 2 != 0:
                        hell_rect = pygame.Rect(col * cell_size,row * cell_size,cell_size,cell_size)
                        pygame.draw.rect(screen,hell_color,hell_rect)

    def draw_score(self):
        #size of the snake body
        score_text = str(len(self.snake.body) - 3)
        #where the score is placed with a black lined box.
        score_surface = game_font.render(score_text,True,(56,74,12))
        score_x = int(cell_size * cell_number - 60)
        score_y = int(cell_size * cell_number - 40)
        score_rect = score_surface.get_rect(center = (score_x,score_y))
        sheep_rect = sheep.get_rect(midright = (score_rect.left,score_rect.centery))
        bg_rect = pygame.Rect(sheep_rect.left,sheep_rect.top,sheep_rect.width + score_rect.width +
6,sheep_rect.height)

        pygame.draw.rect(screen,(167,209,61),bg_rect)
        screen.blit(score_surface,score_rect)
        screen.blit(sheep,sheep_rect)
        pygame.draw.rect(screen,(56,74,12),bg_rect,2)

pygame.mixer.pre_init(44100,-16,2,512)
pygame.init()
cell_size = 40
cell_number = 20
screen = pygame.display.set_mode((cell_number * cell_size,cell_number * cell_size))
clock = pygame.time.Clock()
sheep = pygame.image.load('/Users/taehoonkim/Downloads/3_Pygame/assets/sheep.png').convert_alpha()
game_font = pygame.font.Font('/Users/taehoonkim/Downloads/3_Pygame/assets/PoetsenOne-Regular.ttf', 25)

SCREEN_UPDATE = pygame.USEREVENT
pygame.time.set_timer(SCREEN_UPDATE,150)

main_game = MAIN()

while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()
        if event.type == SCREEN_UPDATE:
            main_game.update()
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP:
                if main_game.snake.direction.y != 1:
                    main_game.snake.direction = Vector2(0,-1)
            if event.key == pygame.K_RIGHT:
                if main_game.snake.direction.x != -1:
                    main_game.snake.direction = Vector2(1,0)
            if event.key == pygame.K_DOWN:
                if main_game.snake.direction.y != -1:
                    main_game.snake.direction = Vector2(0,1)
            if event.key == pygame.K_LEFT:
                if main_game.snake.direction.x != 1:
                    main_game.snake.direction = Vector2(-1,0)

    #filled of the screen
    screen.fill((175,215,70))
    main_game.draw_elements()
    pygame.display.update()
    #the tick is the number of frames per tick.
    clock.tick(60)
```

**The Result of the game:**



**Explanation of the results:**

1. **The checkboard of dark green and red is to confuse the player. So, when the snake becomes bigger it will be a lot harder. So no need for speed.**
2. **The Score at the bottom right shows the amount of sheep's eaten**

3. **Ironically the sound of the eating is rather pleasant as it is a “crunching sound” form the game “Minecraft”**