Code

```
import pygame
import time
import random

# Initialize pygame
pygame.init()

# Sound effects for eating and game over
pygame.mixer.init()
eat_sound = pygame.mixer.Sound('eat.mp3')
game_over_sound = pygame.mixer.Sound('game_over.mp3')
sad_game_over_sound = pygame.mixer.Sound('sad.mp3')
level_cleared_sound = pygame.mixer.Sound('level_cleared.mp3')

# Game Variables
snake_speed = 5
window_x = 720
window_y = 480

# Defining colors
sea = pygame.Color(72,209, 204)
brown = pygame.Color(210, 105, 30)
white = pygame.Color(255, 255, 255)
red = pygame.Color(255, 0, 0)
green = pygame.Color(0, 100, 0)
blue = pygame.Color(0, 0, 255)

# Initialize game window
pygame.display.set_caption('Snake Game')
game_window = pygame.display.set_mode((window_x, window_y))

# FPS (frames per second) controller
fps = pygame.time.Clock()
```

```
# Snake default position and body
snake_position = [100, 50]
snake_body = [[100, 50], [90, 50], [80, 50], [70, 50]]

# Food position
fruit_position = [random.randrange(1, (window_x // 10)) * 10,
random.randrange(1, (window_y // 10)) * 10]
fruit_spawn = True

# Default direction
direction = 'RIGHT'
change_to = direction

# Score
paused= False
score = 0
level = 1
level_threshold = 50  # Points needed to reach the next level

def show_pause_message():
    pause_font = pygame.font.SysFont('times new roman', 30)
    pause_surface = pause_font.render('Game Paused! Press SPACE
to Continue', True, white)
    pause_rect = pause_surface.get_rect()
    pause_rect.center = (window_x / 2, window_y / 2)
    game_window.blit(pause_surface, pause_rect)

# Function to display score and level
def show_score_and_level(color, font, size):
    score_font = pygame.font.SysFont(font, size)
    score_surface = score_font.render('Score: ' + str(score) + '
Level: ' + str(level), True, color)
    score_rect = score_surface.get_rect()
```

```
    score_rect.midtop = (window_x / 2, 15)
    game_window.blit(score_surface, score_rect)

# Function to display level cleared message
def show_level_cleared_message(level):
    level_cleared_sound.play()
    my_font = pygame.font.SysFont('times new roman', 50)
    level_surface = my_font.render('Level ' + str(level) + '
Cleared!', True, white)
    level_rect = level_surface.get_rect()
    level_rect.midtop = (window_x / 2, window_y / 4)
    game_window.blit(level_surface, level_rect)
    pygame.display.flip()
    time.sleep(5)  # Pause for 2 seconds to show the message


# Game over function
def game_over():
    if score <= 30:
        sad_game_over_sound.play()
    else:
        game_over_sound.play()  # Play game over sound

    my_font = pygame.font.SysFont('times new roman', 60)

    # Render the score and level separately
    game_over_score_surface = my_font.render('Your Score is: ' +
str(score), True, green)
    game_over_level_surface = my_font.render('You are at LEVEL :
' + str(level), True, red)

    # Get the rectangles for score and level
    game_over_score_rect = game_over_score_surface.get_rect()
    game_over_level_rect = game_over_level_surface.get_rect()
```

```
    # Set positions for the score and level text
    game_over_score_rect.midtop = (window_x / 2, window_y / 4)
    game_over_level_rect.midtop = (window_x / 2, window_y / 4 +
50)  # Adjust position for the level below the score

    # Display score and level on the screen
    game_window.blit(game_over_score_surface,
game_over_score_rect)
    game_window.blit(game_over_level_surface,
game_over_level_rect)

    # Update the display to show the changes
    pygame.display.flip()

    # Pause for a while to let the player see the message
    time.sleep(5)
    pygame.quit()
    quit()

# Main loop
while True:
    # Event handling
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            game_over()  # End the game when the close button is
clicked
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_UP and direction != 'DOWN':
                change_to = 'UP'
            elif event.key == pygame.K_DOWN and direction !=
'UP':
                change_to = 'DOWN'
```

```
            elif event.key == pygame.K_LEFT and direction !=
'RIGHT':
                change_to = 'LEFT'
            elif event.key == pygame.K_RIGHT and direction !=
'LEFT':
                change_to = 'RIGHT'
            elif event.key == pygame.K_SPACE:  # Pause the game
                paused = not paused
    if not paused:
        # Updating direction
        direction = change_to

        # Snake movement
        if direction == 'UP':
            snake_position[1] -= 10
        elif direction == 'DOWN':
            snake_position[1] += 10
        elif direction == 'LEFT':
            snake_position[0] -= 10
        elif direction == 'RIGHT':
            snake_position[0] += 10

        # Snake growing mechanism and checking if it eats food
        snake_body.insert(0, list(snake_position))
        if snake_position[0] == fruit_position[0] and
snake_position[1] == fruit_position[1]:
            score += 10
            snake_speed += 2  # Increase speed when snake eats
food
            eat_sound.play()  # Play eating sound
            fruit_spawn = False
        else:
            snake_body.pop()
```

```
        # Spawning new food
        if not fruit_spawn:
            fruit_position = [random.randrange(1, (window_x //
10)) * 10, random.randrange(1, (window_y // 10)) * 10]
        fruit_spawn = True

        # Filling the background
        game_window.fill(sea)

        # Drawing the snake
        for pos in snake_body:
            pygame.draw.rect(game_window, green,
pygame.Rect(pos[0], pos[1], 10, 10))

         # Drawing the food
        pygame.draw.rect(game_window, brown,
pygame.Rect(fruit_position[0], fruit_position[1], 10, 10))

        # Game Over conditions
        if snake_position[0] < 0 or snake_position[0] > window_x
- 10 or snake_position[1] < 0 or snake_position[1] > window_y -
10:
            game_over()

        # Check if snake collides with itself
        for block in snake_body[1:]:
            if snake_position[0] == block[0] and snake_position[1]
== block[1]:
                game_over()

        # Check if the level is cleared
        if score >= level * level_threshold:
            show_level_cleared_message(level)
            level += 1  # Move to the next level
```

```
        # Displaying the score and level
    show_score_and_level(white, 'times new roman', 30)
else:
    # Show pause message
    show_pause_message()


# Refresh game screen
pygame.display.update()

# Frame Per Second / Refresh Rate
fps.tick(snake_speed)
```