PYGAME作品

--貪吃蛇小遊戲--

2024/5/19

報告者:陳泓睿

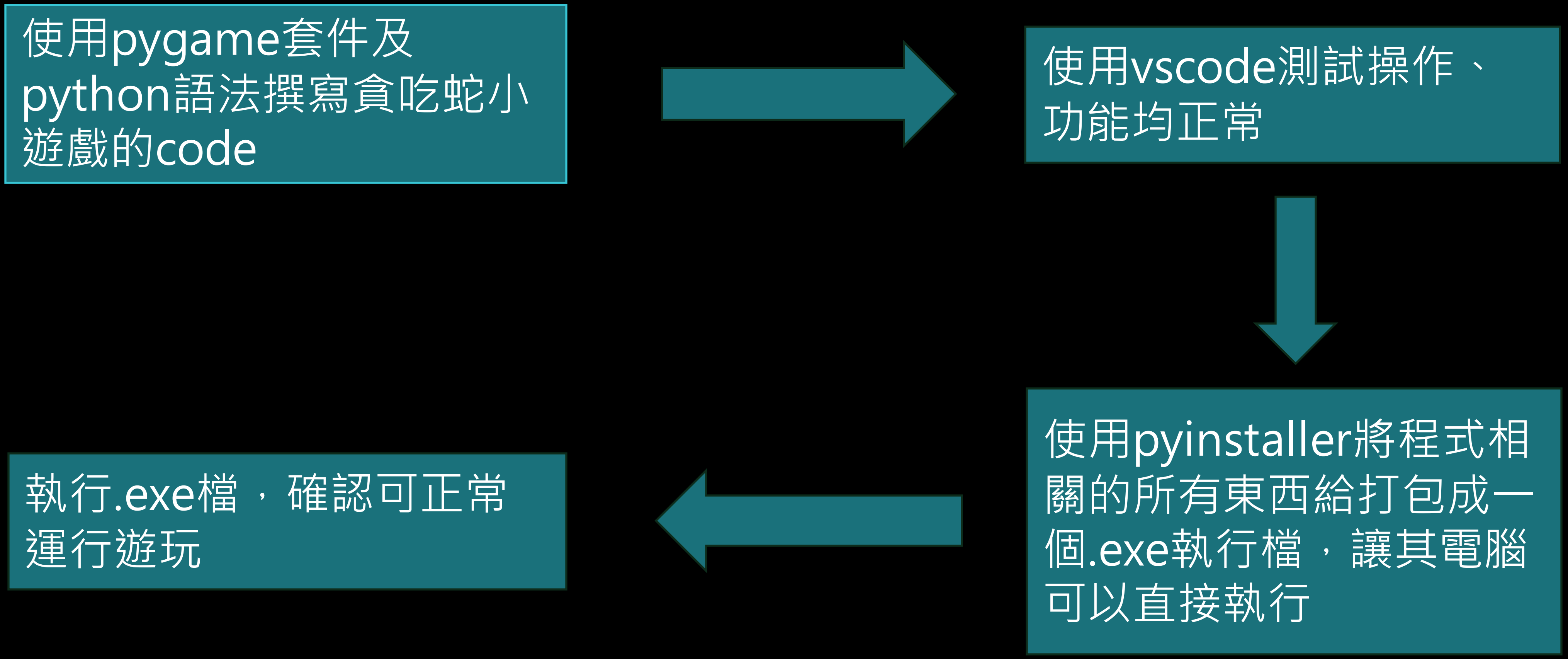
目錄

- 簡介
- 流程圖
- 工具
- 作品展示

簡介

製作的貪食蛇小遊戲，可在進入遊戲時先選擇難易度，並可在遊戲中控制一條蛇向食物移動吃掉並成長，隨著吃掉的食物越多分數越高，可以透過鍵盤控制蛇的移動，讓您在休閒時間享受小時候的懷舊小遊戲！

製作流程圖



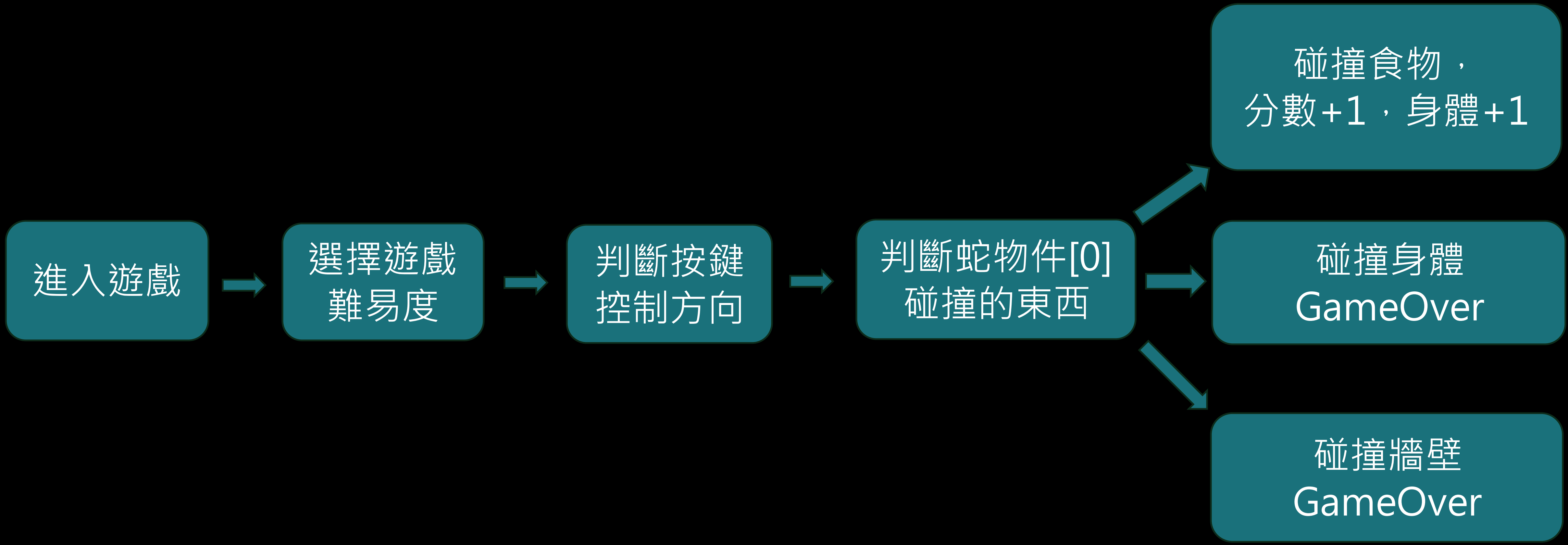
使用pygame套件及python語法撰寫貪吃蛇小遊戲的code

使用vscode測試操作、功能均正常

使用pyinstaller將程式相關的所有東西給打包成一個.exe執行檔，讓其電腦可以直接執行

執行.exe檔，確認可正常運行遊玩

邏輯流程圖



進入遊戲

選擇遊戲難易度

判斷按鍵控制方向

判斷蛇物件[0]碰撞的東西

碰撞食物，分數+1，身體+1

碰撞身體GameOver

碰撞牆壁GameOver

使用工具

使用vscode撰寫code及程式執行測試

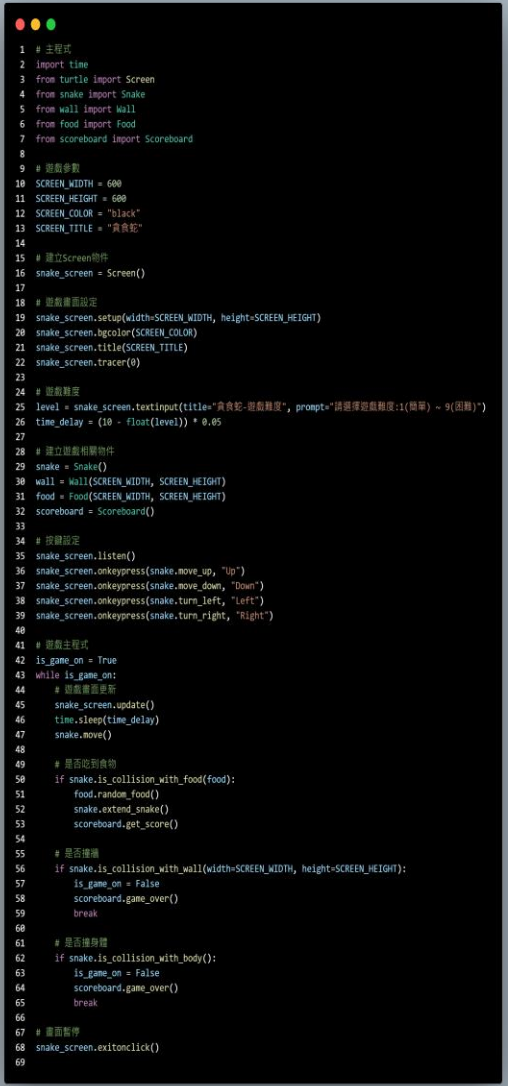
使用pyinstaller將程式相關的所有東西給打包成一個.exe執行檔

使用MySql將分數資料存放在Database

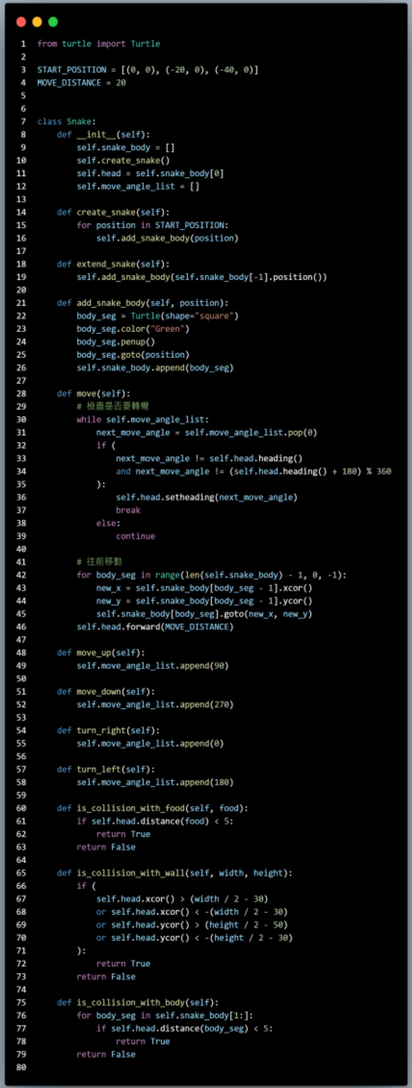
使用PyGame套件

使用Python語言撰寫相關程式碼

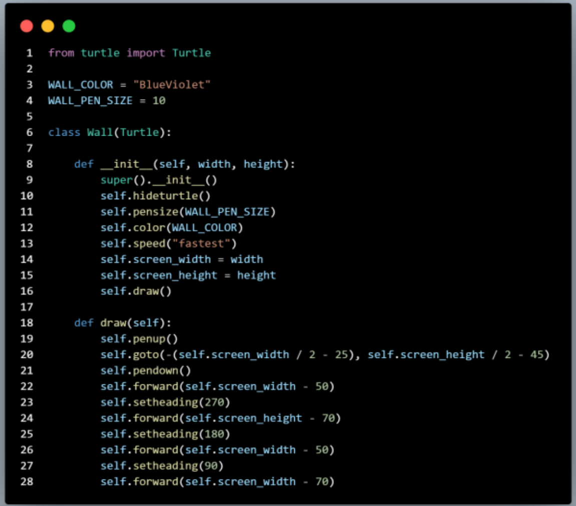
作品展示



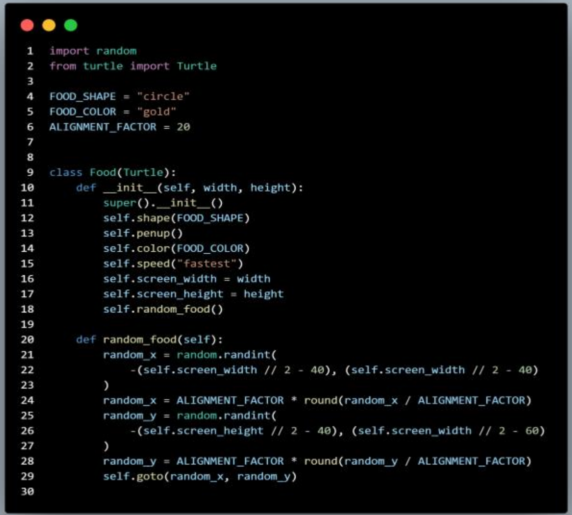
```python
# 主程式
import time
from turtle import Screen
from snake import Snake
from wall import Wall
from food import Food
from scoreboard import Scoreboard

# 遊戲參數
SCREEN_WIDTH = 600
SCREEN_HEIGHT = 600
SCREEN_COLOR = "black"
SCREEN_TITLE = "貪食蛇"

# 建立Screen物件
snake_screen = Screen()

# 遊戲畫面設定
snake_screen.setup(width=SCREEN_WIDTH, height=SCREEN_HEIGHT)
snake_screen.bgcolor(SCREEN_COLOR)
snake_screen.title(SCREEN_TITLE)
snake_screen.tracer(0)

# 遊戲難度
level = snake_screen.textinput(title="貪食蛇-遊戲難度", prompt="請選擇遊戲難度:1(簡單) ~ 9(困難)")
time_delay = (10 - float(level)) * 0.05

# 建立遊戲相關物件
snake = Snake()
wall = Wall(SCREEN_WIDTH, SCREEN_HEIGHT)
food = Food(SCREEN_WIDTH, SCREEN_HEIGHT)
scoreboard = Scoreboard()

# 按鍵設定
snake_screen.listen()
snake_screen.onkeypress(snake.move_up, "Up")
snake_screen.onkeypress(snake.move_down, "Down")
snake_screen.onkeypress(snake.turn_left, "Left")
snake_screen.onkeypress(snake.turn_right, "Right")

# 遊戲主程式
is_game_on = True
while is_game_on:
    # 遊戲畫面更新
    snake_screen.update()
    time.sleep(time_delay)
    snake.move()

    # 是否吃到食物
    if snake.is_collision_with_food(food):
        food.random_food()
        snake.extend_snake()
        scoreboard.get_score()

    # 是否撞牆
    if snake.is_collision_with_wall(width=SCREEN_WIDTH, height=SCREEN_HEIGHT):
        is_game_on = False
        scoreboard.game_over()
        break

    # 是否撞身體
    if snake.is_collision_with_body():
        is_game_on = False
        scoreboard.game_over()
        break

# 畫面暫停
snake_screen.exitonclick()

```



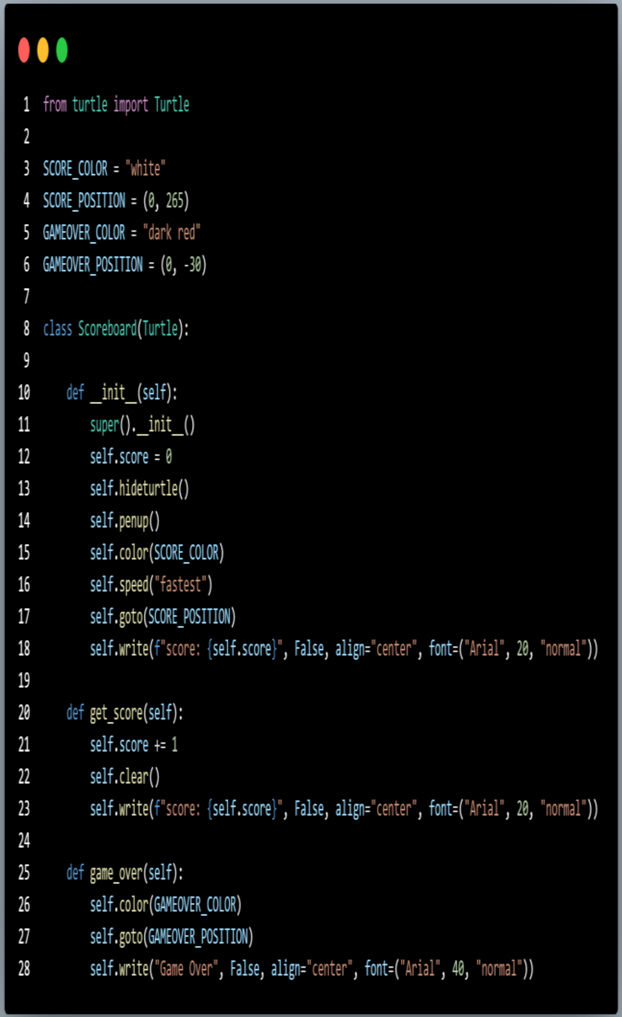
```python
from turtle import Turtle

START_POSITION = [(0, 0), (-20, 0), (-40, 0)]
MOVE_DISTANCE = 20


class Snake:
    def __init__(self):
        self.snake_body = []
        self.create_snake()
        self.head = self.snake_body[0]
        self.move_angle_list = []

    def create_snake(self):
        for position in START_POSITION:
            self.add_snake_body(position)

    def extend_snake(self):
        self.add_snake_body(self.snake_body[-1].position())

    def add_snake_body(self, position):
        body_seg = Turtle(shape="square")
        body_seg.color("Green")
        body_seg.penup()
        body_seg.goto(position)
        self.snake_body.append(body_seg)

    def move(self):
        # 檢查是否要轉彎
        while self.move_angle_list:
            next_move_angle = self.move_angle_list.pop(0)
            if (
                next_move_angle != self.head.heading()
                and next_move_angle != (self.head.heading() + 180) % 360
            ):
                self.head.setheading(next_move_angle)
                break
            else:
                continue

        # 往前移動
        for body_seg in range(len(self.snake_body) - 1, 0, -1):
            new_x = self.snake_body[body_seg - 1].xcor()
            new_y = self.snake_body[body_seg - 1].ycor()
            self.snake_body[body_seg].goto(new_x, new_y)
        self.head.forward(MOVE_DISTANCE)

    def move_up(self):
        self.move_angle_list.append(90)

    def move_down(self):
        self.move_angle_list.append(270)

    def turn_right(self):
        self.move_angle_list.append(0)

    def turn_left(self):
        self.move_angle_list.append(180)

    def is_collision_with_food(self, food):
        if self.head.distance(food) < 5:
            return True
        return False

    def is_collision_with_wall(self, width, height):
        if (
            self.head.xcor() > (width / 2 - 30)
            or self.head.xcor() < -(width / 2 - 30)
            or self.head.ycor() > (height / 2 - 50)
            or self.head.ycor() < -(height / 2 - 30)
        ):
            return True
        return False

    def is_collision_with_body(self):
        for body_seg in self.snake_body[1:]:
            if self.head.distance(body_seg) < 5:
                return True
        return False

```



```python
from turtle import Turtle

WALL_COLOR = "BlueViolet"
WALL_PEN_SIZE = 10

class Wall(Turtle):

    def __init__(self, width, height):
        super().__init__()
        self.hideturtle()
        self.pensize(WALL_PEN_SIZE)
        self.color(WALL_COLOR)
        self.speed("fastest")
        self.screen_width = width
        self.screen_height = height
        self.draw()

    def draw(self):
        self.penup()
        self.goto(-(self.screen_width / 2 - 25), self.screen_height / 2 - 45)
        self.pendown()
        self.forward(self.screen_width - 50)
        self.setheading(270)
        self.forward(self.screen_height - 70)
        self.setheading(180)
        self.forward(self.screen_width - 50)
        self.setheading(90)
        self.forward(self.screen_width - 70)
```

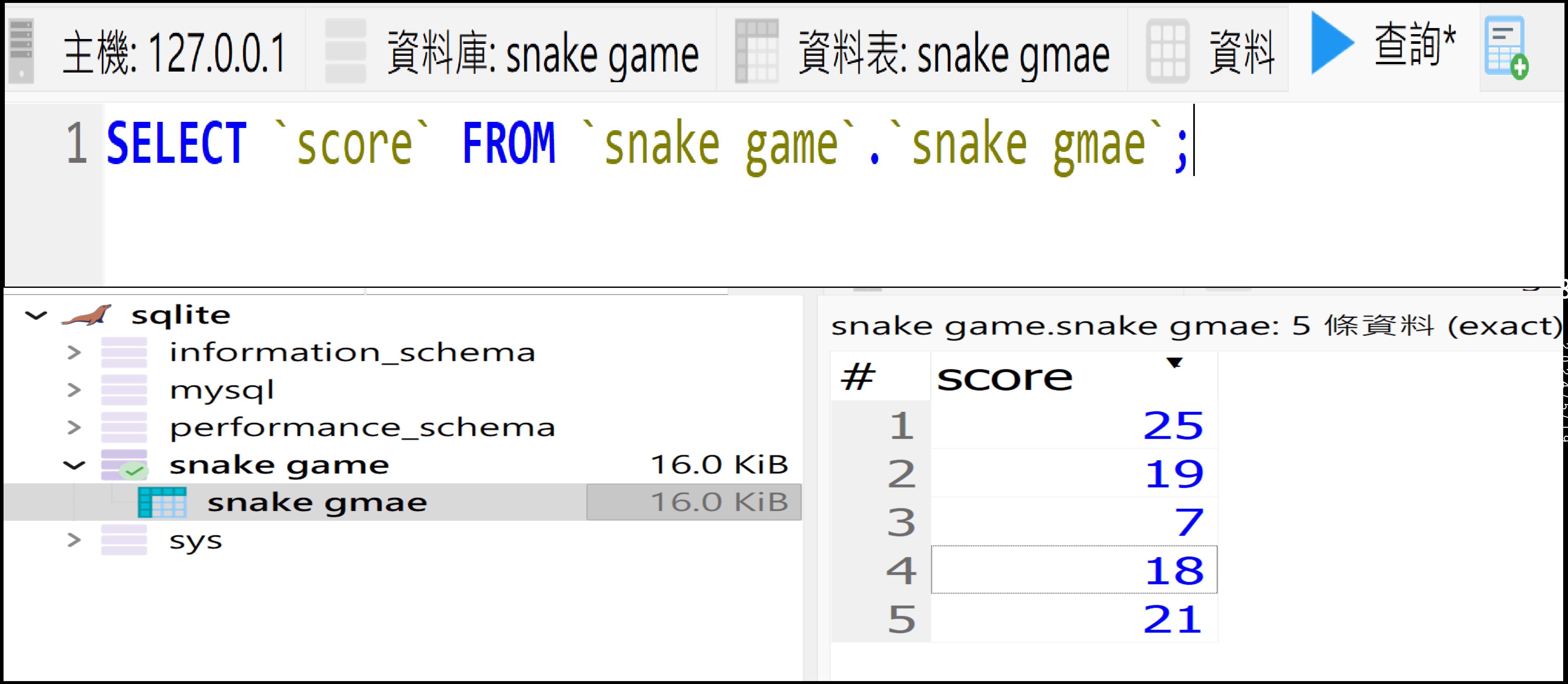


```python
import random
from turtle import Turtle

FOOD_SHAPE = "circle"
FOOD_COLOR = "gold"
ALIGNMENT_FACTOR = 20


class Food(Turtle):
    def __init__(self, width, height):
        super().__init__()
        self.shape(FOOD_SHAPE)
        self.penup()
        self.color(FOOD_COLOR)
        self.speed("fastest")
        self.screen_width = width
        self.screen_height = height
        self.random_food()

    def random_food(self):
        random_x = random.randint(
            -(self.screen_width // 2 - 40), (self.screen_width // 2 - 40)
        )
        random_x = ALIGNMENT_FACTOR * round(random_x / ALIGNMENT_FACTOR)
        random_y = random.randint(
            -(self.screen_height // 2 - 40), (self.screen_width // 2 - 60)
        )
        random_y = ALIGNMENT_FACTOR * round(random_y / ALIGNMENT_FACTOR)
        self.goto(random_x, random_y)

```



```python
from turtle import Turtle

SCORE_COLOR = "white"
SCORE_POSITION = (0, 265)
GAMEOVER_COLOR = "dark red"
GAMEOVER_POSITION = (0, -30)

class Scoreboard(Turtle):

    def __init__(self):
        super().__init__()
        self.score = 0
        self.hideturtle()
        self.penup()
        self.color(SCORE_COLOR)
        self.speed("fastest")
        self.goto(SCORE_POSITION)
        self.write(f"score: {self.score}", False, align="center", font=("Arial", 20, "normal"))

    def get_score(self):
        self.score += 1
        self.clear()
        self.write(f"score: {self.score}", False, align="center", font=("Arial", 20, "normal"))

    def game_over(self):
        self.color(GAMEOVER_COLOR)
        self.goto(GAMEOVER_POSITION)
        self.write("Game Over", False, align="center", font=("Arial", 40, "normal"))
```

作品展示

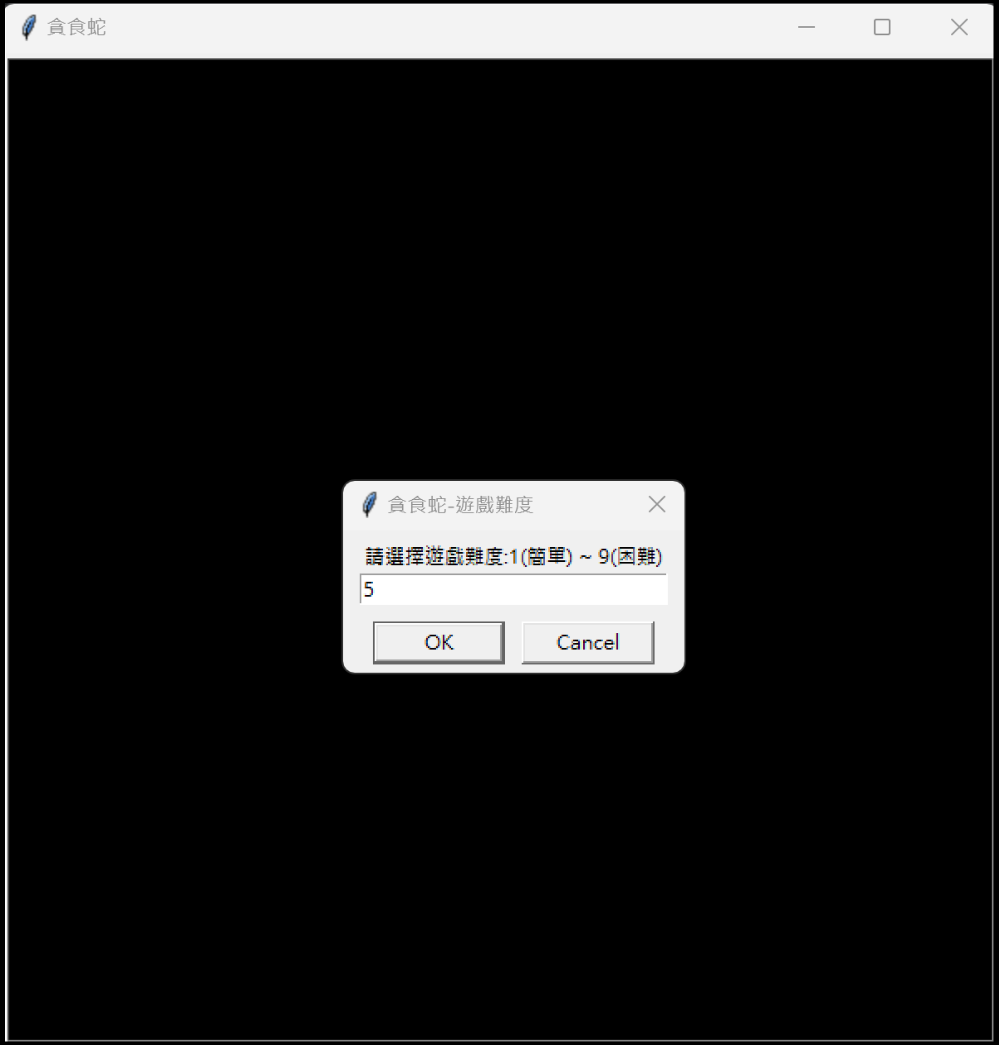


```
SELECT `score` FROM `snake game`.`snake gmae`;
```

snake game.snake gmae: 5 條資料 (exact)

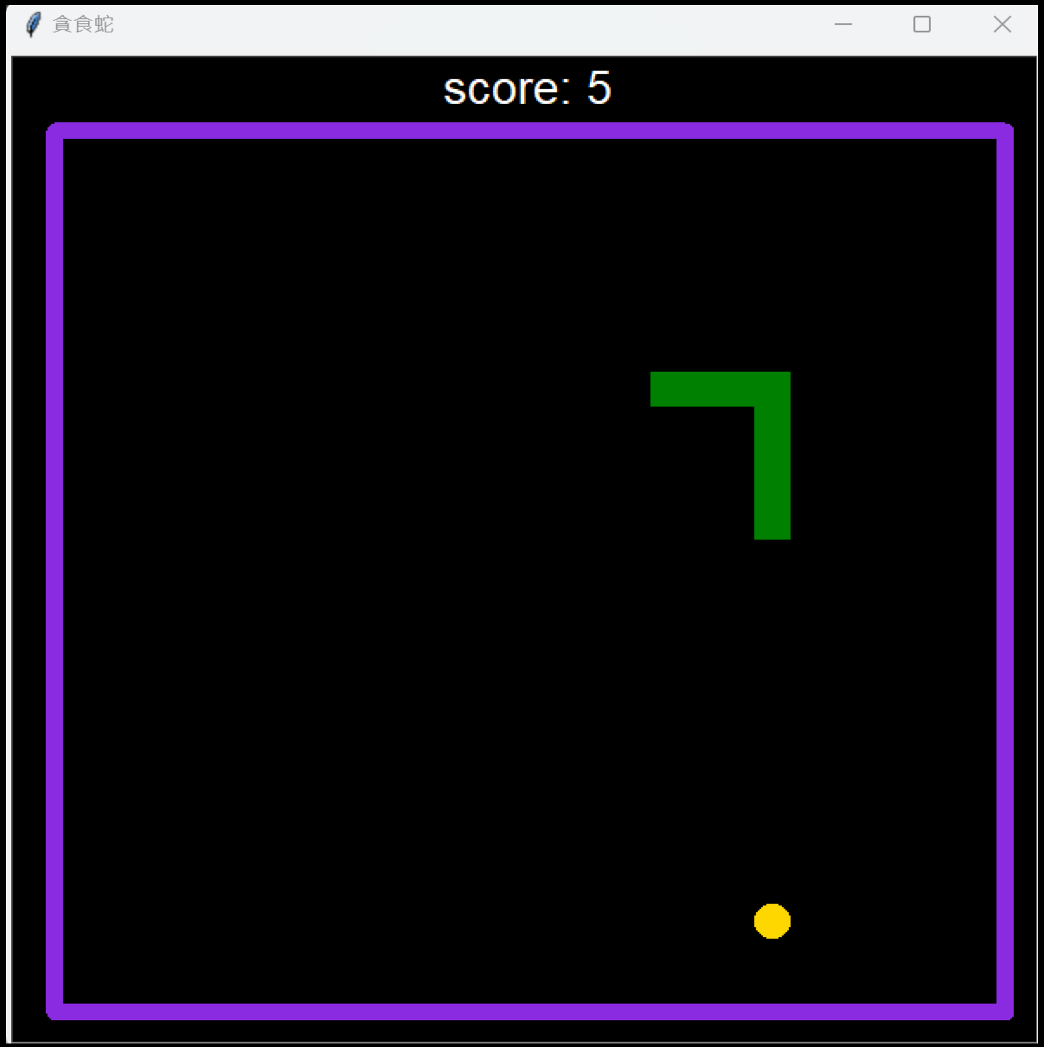
#	score
1	25
2	19
3	7
4	18
5	21

作品展示



請選擇遊戲難度:1(簡單) ~ 9(困難)

5



score: 5