ΒΑΣΙΛΗΣ ΔΕΜΕΡΤΖΗΣ

Φιδάκι (Snake) για Thonny — Python (Tkinter)

Περιγραφή

Αυτό το αρχείο περιέχει τον πλήρη κώδικα για το κλασικό παιχνίδι 'Φιδάκι' (Snake) γραμμένο σε Python με χρήση του Tkinter,

έτοιμο για εκτέλεση στο περιβάλλον Thonny. Περιλαμβάνει οδηγίες για εκτέλεση και μικρές προσαρμογές.

Απαιτήσεις

- Thonny (συνήθως συνοδεύει Python και Tkinter).
- Python 3.x.
- Δεν απαιτούνται επιπλέον βιβλιοθήκες.

Οδηγίες

---

1. Άνοιξε το Thonny.
2. Δημιούργησε νέο αρχείο (File → New).
3. Αντέγραψε τον κώδικα που ακολουθεί και επικόλλησέ τον στο αρχείο.
4. Αποθήκευσε το αρχείο ως snake_thonny.py.
5. Πάτησε Run → Run current script (ή F5) για να ξεκινήσει το παιχνίδι.

6. Χρήση: Τα βελάκια (← ↑ → ↓) ελέγχουν την κίνηση του φιδιού. Το παιχνίδι τελειώνει αν το φίδι προσκρούσει στα τοιχώματα ή στον εαυτό του.

Σημειώσεις βελτίωσης

- Μπορείτε να αλλάξετε το μέγεθος του καμβά, την ταχύτητα (μεταβλητή `delay`) ή τα χρώματα.

- Για καλύτερη εμπειρία/ανάλυση, μεγάλωσε το μέγεθος κελιού (cell_size) ή μείωσε το πλάτος/ύψος σε κελία.

Κώδικας (επικόλλησε σε αρχείο .py στο Thonny)

```
# Snake game for Thonny (Python + Tkinter)
# Απλά επικόλλησε αυτό το αρχείο στο Thonny και τρέξε το.
import tkinter as tk
import random

# Ρυθμίσεις
CELL_SIZE = 20    # pixels per cell
GRID_WIDTH = 30   # number of cells horizontally
GRID_HEIGHT = 20  # number of cells vertically
DELAY = 100       # milliseconds between frames (μικρότερο -> πιο γρήγορο)

class SnakeGame:
    def __init__(self, master):
        self.master = master
        self.width = CELL_SIZE * GRID_WIDTH
        self.height = CELL_SIZE * GRID_HEIGHT
        master.title("Φιδάκι - Snake (Thonny)")

        self.canvas = tk.Canvas(master, width=self.width, height=self.height, bg="black")
        self.canvas.pack()

        self.reset_game()
        master.bind("<Key>", self.on_key)
        self.running = True
        self.after_id = None
        self.game_loop()

    def reset_game(self):
        # αρχικό φίδι - τρία κομμάτια, κινούμενο δεξιά
        self.direction = "Right"
        start_x = GRID_WIDTH // 2
```

```
        start_y = GRID_HEIGHT // 2
        self.snake = [(start_x-1, start_y), (start_x, start_y),
(start_x+1, start_y)]
        self.spawn_food()
        self.score = 0
        self.draw()

    def spawn_food(self):
        while True:
            fx = random.randint(0, GRID_WIDTH-1)
            fy = random.randint(0, GRID_HEIGHT-1)
            if (fx, fy) not in self.snake:
                self.food = (fx, fy)
                break

    def on_key(self, event):
        key = event.keysym
        # Αποτροπή αντίστροφης κίνησης
        opposites = {"Left":"Right", "Right":"Left", "Up":"Down",
"Down":"Up"}
        mapping = {"Left":"Left", "Right":"Right", "Up":"Up",
"Down":"Down",
                   "w":"Up", "a":"Left", "s":"Down", "d":"Right"}
        if key in mapping:
            new_dir = mapping[key]
            if opposites.get(new_dir) != self.direction:
                self.direction = new_dir

    def game_loop(self):
        if not self.running:
            return
        self.update()
        self.draw()
        self.after_id = self.master.after(DELAY, self.game_loop)

    def update(self):
        head_x, head_y = self.snake[-1]
        if self.direction == "Left":
            head_x -= 1
        elif self.direction == "Right":
            head_x += 1
        elif self.direction == "Up":
            head_y -= 1
        elif self.direction == "Down":
            head_y += 1

        new_head = (head_x, head_y)

        # Έλεγχος σύγκρουσης με τοίχους
```

```
        if not (0 <= head_x < GRID_WIDTH and 0 <= head_y <
GRID_HEIGHT):
            self.game_over()
            return

        # Έλεγχος σύγκρουσης με το ίδιο το σώμα
        if new_head in self.snake:
            self.game_over()
            return

        # Προσθήκη νέας κεφαλής
        self.snake.append(new_head)

        # Έλεγχος αν έφαγε τροφή
        if new_head == self.food:
            self.score += 1
            self.spawn_food()
        else:
            # αφαίρεση ουράς για να κινηθεί
            self.snake.pop(0)

    def draw(self):
        self.canvas.delete("all")

        # Ζωγραφίζουμε τροφή
        fx, fy = self.food
        self.draw_cell(fx, fy, fill="red")

        # Ζωγραφίζουμε φίδι
        for i, (x, y) in enumerate(self.snake):
            if i == len(self.snake) - 1:
                # κεφαλή
                self.draw_cell(x, y, fill="yellow")
            else:
                # σώμα
                self.draw_cell(x, y, fill="green")

        # Σκορ
        self.canvas.create_text(10, 10, anchor="nw", fill="white",
text=f"Score: {self.score}")

    def draw_cell(self, grid_x, grid_y, fill="white"):
        x1 = grid_x * CELL_SIZE
        y1 = grid_y * CELL_SIZE
        x2 = x1 + CELL_SIZE
        y2 = y1 + CELL_SIZE
        self.canvas.create_rectangle(x1, y1, x2, y2, outline="",
fill=fill)
```

```
    def game_over(self):
        self.running = False
        if self.after_id:
            self.master.after_cancel(self.after_id)
        self.canvas.create_text(self.width//2, self.height//2-10,
text="GAME OVER", fill="white", font=("Arial", 24))
        self.canvas.create_text(self.width//2, self.height//2+20,
text=f"Final Score: {self.score}", fill="white", font=("Arial", 16))
        self.canvas.create_text(self.width//2, self.height//2+60,
text="Press R to restart", fill="white", font=("Arial", 12))
        self.master.bind("r", lambda e: self.restart())

    def restart(self):
        self.reset_game()
        self.running = True
        self.game_loop()

if __name__ == "__main__":
    root = tk.Tk()
    game = SnakeGame(root)
    root.mainloop()
```