Sprawozdanie

WSB Merito

Programista Python Developer – grupa 1

Joanna Banachowicz

1) Wstęp

W ramach projektu zdecydowałam się stworzyć własną wersję gry ‘Snake’. Gra posiada pięć poziomów trudności. Stworzyłam także interfejs, który umożliwia wybór poziomu, klawiszy sterujących oraz sprawdzenie najlepszych graczy. Grę oparłam o bibliotekę pygame oraz pygame_gui. Link do repozytorium, w którym jest cały kod gry: https://github.com/JoannaBanach/Snake.git

2) Opis szczegółowy

Gra została podzielona na kilka modułów:

a) Klasa Obstacle

Klasa zawiera funkcjonalności:

- Tworzenia przeszkody w jednej z czterech ćwiartek pola, po którym porusza się wąż.

- Wylosowanie miejsca, gdzie znajduje się przeszkoda oraz wylosowanie jej wielkości

- Przekazanie współrzędnych przeszkody.

- Rysowanie przeszkody.

Fragment kodu:

```
class Obstacle:
    def __init__(self, quarter, background):
        if quarter == 1:
            self.x = random.randint(2, 17) * 10
            self.y = random.randint(2, 17) * 10
        elif quarter == 2:
            self.x = random.randint(22, 33) * 10
            self.y = random.randint(2, 17) * 10
        elif quarter == 3:
            self.x = random.randint(2, 17) * 10
            self.y = random.randint(22, 33) * 10
        elif quarter == 4:
            self.x = random.randint(22, 33) * 10
            self.y = random.randint(22, 33) * 10

        self.height = random.randint(1, 5) * 10
        self.width = random.randint(1, 5) * 10
        self.color = (128, 128, 128)
        self.background = background
```

b) Klasa Apple

Klasa zawiera funkcjonalności:

- Losowe generowanie, gdzie jabłko będzie się znajdowało.

- Losowanie współrzędnych jabłka obejmuje sprawdzenie, gdzie w obecnym momencie znajduje się wąż oraz przeszkody i inne jabłka, jeśli dany poziom je zawiera.

- Zmiana koloru jabłka z czerwonego na zielone, gdy jego zjedzenie jest premiowane dodatkowymi punktami.

- Przekazywanie współrzędnych jabłka.

- Rysowanie jabłka.

Fragment kodu:

```
class Apple:
    def __init__(self, background, if_green_apple: bool, **kwargs):
        self.x = random.randint(1, 38) * 10
        self.y = random.randint(1, 38) * 10
        self.height = 10
        self.width = 10
        self.color = (255, 0, 0)
        self.background = background
        self.if_green_apple = if_green_apple

        coords_occupied = []
        for arg_name, arg_value in zip(kwargs.keys(), kwargs.values()):
            if arg_name == 'obstacles_coords':
                coords_occupied = [*coords_occupied, *arg_value]
            elif arg_name == 'apples':
                apples_coords = []
                for i in range(len(arg_value)):
                    apples_coords.append(arg_value[i].coord())

        while self.coord() in coords_occupied:
            self.x = random.randint(1, 38) * 10
            self.y = random.randint(1, 38) * 10
```

c) Klasa SaveScore

Klasa zawiera funkcjonalności:

- Sprawdzenie, czy plik zawierający najlepsze wyniki istnieje.

- Utworzenie pliku, jeśli nie istnieje.

- Uzupełnienie brakujących zakładek do każdego poziomu gry.

- Zapisywanie wyniku gry, przypisując go do odpowiedniego poziomu i właściwego zajętego miejsca.

- Zwrócenie zajętego miejsca.

- Sprawdzenie najlepszego wyniku dla każdego poziomu.

Fragment kodu:

```
class SaveScore:
    def __init__(self):
        self.name = "GameSnakeScore.xlsx"
        self.levels = ['Super Easy', 'Easy', 'Medium', 'Hard', 'Super
Hard']

    def initiate_file(self):
        self.create_file()
        self.check_workbooks()

    def check_exist(self):
        result = os.path.exists(self.name)
        return result

    def create_file(self):
        if not self.check_exist():
            workbook = xlsxwriter.Workbook(self.name)
            workbook.close()
```

d) Klasa Snake

Klasa zawiera funkcjonalności:

- Podklasa SnakePart, która zawiera współrzędne pojedynczej części węża.

- Utworzenie węża i jego wyglądu.

- Poruszanie się węża.

- Powiększenie węża.

- Sprawdzenie długości węża.

- Sprawdzenie czy wąż zjadł jabłko lub wpadł na przeszkodę bądź granicę planszy.

- Przekazanie współrzędnych węża.

Fragment kodu:

```
class Snake:

    def __init__(self, background, if_walls: bool, if_green_apple: bool):
        self.x_delta = 10
        self.y_delta = 0
        self.background = background
        self.elements = [SnakePart(200, 200, self.background),
SnakePart(190, 200, self.background)]
        self.if_walls = if_walls
        self.if_green_apple = if_green_apple

    def kill(self):
        del self

    def paint(self):
        for part in self.elements:
            part.paint()

    def increase(self):
        x = self.elements[-1].coord()[0] + self.x_delta
        y = self.elements[-1].coord()[1] + self.y_delta
```

```
        self.elements.append(SnakePart(x, y, self.background))

    def length(self):
        return len(self.elements)
```

e) Klasa SnakeGame

Klasa zawiera funkcjonalności:

- Utworzenie planszy, po której porusza się wąż.

- Wyświetlanie bieżącego wyniku.

- Przyspieszanie prędkości węża na żądanie gracza lub w przypadku poziomu Hard i Super Hard także zwiększanie prędkości dodatkowo co 15 sekund.

- Definiowanie ile jabłek i przeszkód będzie zawierać poziom gry oraz czy będą pojawiać się zielone jabłka specjalne.

- Wyświetlanie obecnego stanu gry: położenia jabłek, węża i przeszkód.

Fragment kodu:

```
import pygame
import Apple_Class as ac
import Obstacle_Class as oc
import Snake as se
import threading


class SnakeGame:
    def __init__(self, controls_dict, level):
        self.FPS = 5
        self.window_width = 400
        self.window_height = 450
        self.window = (self.window_width, self.window_height)
        self.done = False
        self.score = 0
        self.controls_dict = controls_dict
        self.level = level
        self.area = (0, 0, 400, 400)
        self.FPS_acc = 0
        self.snake_exists = False

   def print_score(self, screen):
    font = pygame.font.Font(None, 36)
    score_text = font.render(f'Score: {self.score}', True, (255, 255, 255))
    screen.blit(score_text, (10, 410))
```

f) Klasa SnakeGUI

Klasa odpowiada za utworzenie trzech interfejsów:

- Interfejs początkowy, który umożliwia:

i) wprowadzenie imienia lub pseudonimu gracza,

i) wybór klawiszy sterujących,

i) wybór poziomu gry,

i) wywołanie sprawdzenia najlepszych wyników dla każdego poziomu,

i) rozpoczęcie gry.

- Interfejs wyświetlający najlepsze wyniki dla każdego poziomu. Interfejs zawiera przycisk umożliwiający powrót do głównego interfejsu.

- Interfejs informujący o zakończeniu gry. Wyświetlana jest zdobyta ilość punktów oraz zajęte miejsce, pod warunkiem, że zajęło się jedno z pierwszych dziesięciu miejsc. Interfejs zawiera przycisk umożliwiający powrót do głównego interfejsu.

Do klasy zostały utworzone dodatkowe pliki z ustawieniami, jak powinny wyglądać na interfejsach pola tekstowe, przyciski i listy wybieralne. Dzięki temu wygląd wszystkich interfejsów jest spójny.

Fragment kodu:

```
class SnakeGUI:
    def __init__(self):
        self.window_width = 600
        self.window_height = 400
        self.window = (self.window_width, self.window_height)
        self.done = False
        self.level = 'Easy'
        self.player_name = 'Nickname'
        self.controls = 'Arrows'
        self.controls_dict = {}
        self.run_game = False

    def start_game(self):
        if self.level in ['Super Easy', 'Easy', 'Medium', 'Hard', 'Super
Hard']:
            snake = sg.SnakeGame(self.controls_dict, self.level)
        else:
            exit()
        score = snake.start_game()
        del snake
        self.end_window(score)
```

Fragment ustawienia wyglądu interfejsu:

```
{
    "text_box":
    {
        "colours":
        {
            "dark_bg":"#C0C0C0",
            "normal_text": "#000000"
        },

        "misc":
        {
           "border_width": "0",
           "shadow_width": "0",
           "text_horiz_alignment": "center"
        }
    }
}
```

3) Podsumowanie gry

Gra posiada zdefiniowane trzy poziomy:

- Super Easy - nie posiada granicy planszy (wąż pojawia się po przeciwnej stronie). Na planszy jest umieszczonych 5 jabłek.

- Easy – także nie posiada granicy planszy, natomiast na planszy jest jedno jabłko.

- Medium – zdefiniowane granice planszy. Na planszy jest jedno jabłko, które okazjonalnie staje się premiowanym zielonym jabłkiem.

- Hard – zdefiniowane granice planszy oraz cztery przeszkody. Na planszy są trzy jabłka, z czego jedno okazjonalnie staje się premiowanym zielonym jabłkiem. Prędkość węża jest zwiększa co 15 sekund o 1.

- Super Hard – zdefiniowane granice planszy oraz cztery przeszkody Na planszy jest jedno jabłko, które okazjonalnie staje się premiowanym zielonym jabłkiem. Prędkość węża jest mnożona co 15 sekund razy 1.2.

Interfejsy użytkownika są spójne pod względem wyglądu i są intuicyjne w użyciu. Bardzo wyraźnie widać, gdzie na planszy znajduje się wąż. Jabłko i przeszkody można odróżnić poprzez odmienne kolory. Gra zawiera elementy losowości umiejscowienia jabłek i przeszkód. Gracz może przyspieszać prędkość węża przez ponowne naciśnięcie przycisku w kierunku, w którym aktualnie porusza się wąż.

W kodzie zostały użyte klasy, funkcje klas, importowanie modułów wbudowanych oraz z plików, zastosowano kwargsy, pętle for i while oraz konstrukcje warunkowe.

4) Podsumowanie projektu

Projekt okazał się wyzwaniem, gdyż początkowo kod zawierał kilka różnych klas Snake oraz SnakeGame dla każdego poziomu. Utrudniało to wprowadzanie zmian w kodzie. Zdecydowałam się na zredukowanie ilości klas, dzięki czemu kod zyskał dużą elastyczności i kod nie powtarza się w kilku miejscach. Projekt był doskonałą okazją do wypróbowania umiejętności zdobytych w trakcie zajęć oraz integracji wiedzy o poszczególnych bibliotekach. Dodatkowo był okazją do samodzielnego pogłębienia wiedzy o funkcjach i bibliotekach, które nie zostały omówione.