s30724_PROJEKT_1_PPY_DOKUMENTACJA – GAME OF LIFE

LEGENDA:
ROZDZIAŁ A – INSTRUKCJA URUCHOMIENIA I OBSŁUGI PROGRAMU
ROZDZIAŁ B – OPIS ZASADY I PRZEBIEGU TESTOWANIA FUNKCJONALNOŚCI
ROZDZIAŁ C – OPIS PROBLEMÓW
ROZDZIAŁ D – OPIS ZMIENNYCH I FUNKCJI (SPHINX-STYLE DOCSTRING)

ROZDZIAŁ A – INSTRUKCJA URUCHOMIENIA I OBSŁUGI PROGRAMU

WYMAGANIA
- Python 3.x
- Zainstalowany moduł pygame

URUCHOMIENIE PROGRAMU
- W terminalu / IDE należy uruchomić plik gameOfLife.py (całość projektu to 1 plik .py)

OBSŁUGA PROGRAMU
1. Na ekranie startowym:
- Wybierz kolor komórek
- Wybierz wzór początkowy
- Kliknij przycisk START
2. W widoku symulacji:
- Klikaj na komórki, aby je aktywować lub dezaktywować
- Panel sterowania (prawy dolny róg):
- [<] cofa krok
- [>]wykonuje kolejny krok
- CLEAR – czyści siatkę
- [+ / -] zmienia wielkość siatki
- EXP / SIM – włącza/wyłącza tryb symulacji automatycznej

ROZDZIAŁ B – OPIS ZASADY I PRZEBIEGU TESTOWANIA FUNKCJONALNOŚCI

Testowanie przeprowadzono manualnie w trakcie uruchamiania aplikacji:

1. Sprawdzenie siatki:
    - Czy komórki poprawnie się wyświetlają
    - Czy kolory są zgodne z wyborem
2. Interakcje:
    - Czy kliknięcie aktywuje/dezaktywuje komórkę
    - Czy przyciski zmieniają krok, tryb, rozmiar siatki
3. Symulacja:
    - Czy zmiany w siatce są zgodne z regułami gry w życie Conwaya
    - Czy historia działania siatki (cofanie) działa poprawnie

ROZDZIAŁ C – OPIS PROBLEMÓW

- PIERWSZY PROJEKT W PYTHONIE

Był to mój pierwszy projekt napisany w Pythonie, poza samym faktem, że było to dla mnie nowe, nie napotkałem większych problemów, zawsze gdy potrzebowałem jakiejś funkcjonalności, byłem w stanie ją zaimplementować. Kod wyszedł stosunkowo krótki i powstał w 2 wieczory/noce.

ROZDZIAŁ D – OPIS ZMIENNYCH I FUNKCJI (SPHINX-STYLE DOCSTRING)

1. ZMIENNE GLOBALNE

- WIDTH, HEIGHT: wymiary okna gry
- GRID_SIZE: rozmiar jednej komórki
- FPS: liczba klatek na sekundę
- WHITE, BLACK, GRID_COLOR: kolory tła i siatki
- COLOR_NAMES, COLORS: dostępne kolory życia
- PATTERNS: predefiniowane wzory
- selected_color_name, selected_color: wybrany kolor komórek
- selected_pattern: wybrany wzór startowy
- selected_grid_size: rozmiar komórki w grze
- simulation_mode: czy tryb symulacji jest włączony
- start_screen: czy ekran startowy jest widoczny
- history: lista stanów siatki dla cofania kroków

2. FUNKCJE

```
def create_empty_grid(rows, cols):
    """
    Tworzy pustą siatkę o podanych rozmiarach.

    :param rows: liczba wierszy
    :param cols: liczba kolumn
    :return: lista 2D z wartościami 0
    """
```

```
def create_empty_grid(rows, cols):
    """
    Tworzy pustą siatkę o podanych rozmiarach.

    :param rows: liczba wierszy
    :param cols: liczba kolumn
    :return: lista 2D z wartościami 0
    """
```

```
def draw_grid(grid, grid_size, alive_color):
    """
    Rysuje siatkę na ekranie.

    :param grid: dwuwymiarowa lista stanu komórek
    :param grid_size: rozmiar jednej komórki
    :param alive_color: kolor żywych komórek
    """
```

```
def place_pattern(grid, pattern):
    """
    Umieszcza wzór na środku siatki.

    :param grid: siatka gry
    :param pattern: lista współrzędnych komórek wzoru
    """
```

```
def update_grid(grid):
    """
    Aktualizuje siatkę zgodnie z zasadami gry w życie.

    :param grid: obecna siatka
    :return: nowa siatka po jednym kroku symulacji
    """

def draw_button(text, x, y, width, height, selected=False):
    """
    Rysuje przycisk z tekstem.

    :param text: napis przycisku
    :param x: pozycja x
    :param y: pozycja y
    :param width: szerokość
    :param height: wysokość
    :param selected: czy przycisk jest zaznaczony
    """

def draw_overlay(step_counter):
    """
    Rysuje panel sterujący symulacją, wyświetlając aktualny stan gry.

    Panel zawiera informacje pomocnicze takie jak liczba kroków symulacji,
    aktualny tryb działania, przyciski sterujące i inne elementy
interfejsu.

    :param step_counter: Aktualna liczba wykonanych kroków symulacji.
    :type step_counter: int
    """

def handle_start_screen():
    """
    Obsługuje ekran startowy gry, umożliwiający konfigurację początkową.

    Pozwala użytkownikowi wybrać kolor gracza, wzór startowy planszy lub
inne
    ustawienia początkowe. Funkcja renderuje graficzny interfejs wyboru i
    przetwarza dane wejściowe użytkownika (np. kliknięcia myszką).
    """

def game_loop():
    """
    Główna pętla gry, odpowiedzialna za logikę i wizualizację symulacji.

    Funkcja obsługuje rysowanie planszy, reaguje na kliknięcia użytkownika,
    aktualizuje stan symulacji oraz steruje przebiegiem rozgrywki.
    Zawiera główny cykl zdarzeń (event loop) i logikę animacji.
    """

def main():
    """
    Główna funkcja uruchamiająca program.

    Inicjalizuje wszystkie niezbędne komponenty gry, ustawia ekran startowy
    oraz rozpoczyna główną pętlę gry. Punkt wejścia aplikacji.
    """
```