Game of Life - dokumentacja

Jakub Parat

Spis treści

1 Wstęp

Projekt ten jest implementacją automatu komórkowego "Game of Life" stworzonego przez Johna Conwaya. Program został napisany w języku Python z wykorzystaniem biblioteki `Pygame` do wizualizacji oraz `NumPy` i `SciPy` do wydajnych obliczeń macierzowych. Aplikacja umożliwia symulację rozwoju populacji komórek, interakcję z planszą, wstawianie gotowych struktur oraz zapisywanie i wczytywanie stanu gry.

2 Zasady gry

Gra toczy się na dwuwymiarowej siatce. Każda komórka może znajdować się w jednym z dwóch stanów: żywa (1) lub martwa (0). Stan komórki w następnej iteracji zależy od liczby jej żywych sąsiadów:

1. **Przeżycie:** Żywa komórka z 2 lub 3 żywymi sąsiadami pozostaje żywa.
2. **Śmierć:** Żywa komórka z mniej niż 2 sąsiadami umiera (z samotności), a z więcej niż 3 sąsiadami umiera (z przeludnienia).
3. **Narodziny:** Martwa komórka z dokładnie 3 żywymi sąsiadami staje się żywa.

3 Struktura projektu

Projekt składa się z następujących plików:

- `main.py` - Główny plik uruchomieniowy, integrujący menu startowe z oknem gry.
- `logika.py` - Zawiera klasę `Siatka` odpowiedzialną za logikę automatu, obliczanie kolejnych pokoleń (z użyciem splotu macierzy) oraz operacje na danych (zapis/odczyt).
- `widok.py` - Odpowiada za warstwę wizualną (Pygame), rysowanie siatki, obsługę zdarzeń (mysz, klawiatura) i pętlę główną gry.
- `menu_startowe.py` - Implementacja okna konfiguracyjnego opartego na bibliotece `Tkinter`.
- `wzory.py` - Biblioteka wcześniej przygotowanych struktur (np. Szybowiec, Działo Gospera), które można wstawiać na planszę. Biblioteka ta może być stale urozmaicana.
- `ustawienia.py` - Plik konfiguracyjny zawierający stałe (kolory, domyślne wymiary, rozmiar komórki).

4 Jak uruchomić program?

Wymagane jest zainstalowanie bibliotek zewnętrznych:

```
pip install pygame numpy scipy
```

Aby uruchomić program, w folderze z projektem należy wykonać polecenie:

```
python main.py
```

5 Opis klas

5.1 Klasa `Siatka` (plik `logika.py`)

Zarządza stanem planszy i logiką symulacji.

- **Pola instancji:**
  - `int widoczne_kolumny` - Liczba kolumn widocznych na ekranie.
  - `int widoczne_wiersze` - Liczba wierszy widocznych na ekranie.
  - `int rozmiar_komorki` - Rozmiar pojedynczej komórki w pikselach.
  - `int margines` - Dodatkowy obszar poza ekranem (bufor obliczeniowy), domyślnie 10.
  - `int liczba_kolumn` - Całkowita liczba kolumn (widoczne + marginesy).
  - `int liczba_wierszy` - Całkowita liczba wierszy (widoczne + marginesy).
  - `int licznik_czyszczenia` - Licznik pomocniczy do okresowego czyszczenia bufora.
  - `str ograniczanie` - Tryb brzegów planszy ('wrap' dla torusa, 'fill' dla ścian).
  - `numpy.ndarray obecny_stan` - Główna macierz przechowująca stan komórek (0 lub 1).
- **Metody:**
  - `__init__(szerokosc, wysokosc, rozmiar_komorki, tryb_torus)` - Konstruktor klasy, inicjalizuje wymiary i macierz stanu.
  - `generuj_losowa_plansze()` - Wypełnia planszę losowymi komórkami z zadanym prawdopodobieństwem.
  - `oblicz_nastepne_pokolenie()` - Oblicza nowy stan planszy przy użyciu splotu macierzy (`scipy.signal.convolve2d`).
  - `wyczysc_marginesy()` - Zeruje komórki w strefie buforowej (używane w trybie bez torusa).
  - `zmien_stan_komorki(x, y)` - Odwraca stan komórki (żywa/martwa) w podanych współrzędnych.

- `reset_planszy()` - Czyści całą planszę (ustawia wszystkie komórki na 0).
- `wstaw_wzor(x, y, wzor)` - Wstawia podany wzór (macierz) w wybranym miejscu na planszy.
- `zapisz_stan(nazwa_pliku)` - Serializuje obecny stan planszy do pliku binarnego.
- `wczytaj_stan(nazwa_pliku)` - Wczytuje stan planszy z pliku binarnego.

5.2 Klasa `OknoGry` (plik `widok.py`)

Główna klasa odpowiedzialna za wyświetlanie gry i pętlę zdarzeń.

- **Pola instancji:**
  - `int szerokosc` - Szerokość okna w pikselach.
  - `int wysokosc` - Wysokość okna w pikselach.
  - `int fps` - Docelowa liczba klatek na sekundę.
  - `pygame.Surface ekran` - Główna powierzchnia rysowania biblioteki Pygame.
  - `pygame.time.Clock zegar` - Obiekt kontrolujący czas i płynność animacji.
  - `pygame.font.Font czcionka` - Czcionka systemowa do wyświetlania napisów.
  - `int licznik_krokow` - Licznik iteracji symulacji.
  - `list lista_nazw_wzorow` - Lista dostępnych nazw wzorów pobrana z modułu `wzory`.
  - `int indeks_wybranego_wzoru` - Wskaźnik na aktualnie wybrany wzór do wstawienia.
  - `Siatka siatka` - Obiekt logiki gry (instancja klasy `Siatka`).
  - `bool pauza` - Flaga określająca, czy symulacja jest zatrzymana.
  - `bool czy_dziala` - Flaga sterująca główną pętlą programu.
- **Metody:**
  - `__init__(konfiguracja)` - Inicjalizuje okno Pygame i tworzy obiekt siatki na podstawie konfiguracji.
  - `rysuj_info()` - Wyświetla na ekranie informacje (FPS, iteracja, wybrany wzór).
  - `rysuj_siatke()` - Rysuje tło, linie siatki oraz aktywne komórki (kolorowane zależnie od wieku).
  - `zmien_predkosc(wartosc)` - Zmienia limit FPS o zadaną wartość.
  - `obslugiwanie_zdarzen()` - Przetwarza wejście od użytkownika (klawisze, mysz).

- `pobierz_kolor(wiek)` - Oblicza kolor komórki (gradient od zielonego do niebieskiego) na podstawie jej stanu.
- `petla_glowna()` - Główna pętla gry wywołująca logikę i rysowanie w każdej klatce.

5.3 Klasa `MenuKonfiguracyjne` (plik `menu_startowe.py`)

Odpowiada za wyświetlenie okna startowego z ustawieniami.

- **Pola instancji:**
  - `tk.Tk okno_glowne` - Główne okno biblioteki Tkinter.
  - `tk.IntVar wybrana_szerokosc` - Zmienna przechowująca wybraną szerokość okna.
  - `tk.IntVar wybrana_wysokosc` - Zmienna przechowująca wybraną wysokość okna.
  - `tk.BooleanVar start_losowy` - Flaga wyboru losowego stanu początkowego.
  - `tk.BooleanVar tryb_torus` - Flaga wyboru trybu toroidalnego planszy.
  - `tk.IntVar wybrana_predkosc` - Zmienna przechowująca wybraną prędkość początkową.
  - `bool czy_uruchomic_gre` - Flaga informująca, czy użytkownik zatwierdził uruchomienie gry.
- **Metody:**
  - `__init__()` - Tworzy okno Tkinter i inicjalizuje zmienne domyślne.
  - `budowanie_interfejsu()` - Układa widżety (etykiety, pola tekstowe, przyciski) w oknie.
  - `zatwierdz()` - Waliduje wprowadzone dane i zamyka okno menu.
  - `uruchom()` - Uruchamia pętlę zdarzeń Tkinter i zwraca słownik z konfiguracją po zamknięciu.

6 Szczegółowy opis działania programu

Po uruchomieniu aplikacji pojawia się okno konfiguracyjne. Użytkownik ma możliwość wyboru:

- Rozmiaru okna gry,
- Startu z losową planszą,
- Startu gry w trybie Torus (więcej o tym trybie w sekcji 6.2),
- Prędkości w FPS.

Po zatwierdzeniu ustawień (naciśnięciu przycisku "URUCHOM SYMULACJĘ") otwiera się główne okno symulacji.

6.1 Sterowanie i skróty klawiszowe

Program obsługiwany jest za pomocą myszy i klawiatury:

- **Mysz:**
  - **LPM (Lewy Przycisk Myszy):** Ożywienie lub zabicie komórki w miejscu kursora.
  - **PPM (Prawy Przycisk Myszy):** Wstawienie wybranego wzoru (np. Szybowca) w miejscu kursora.
- **Klawiatura - Symulacja:**
  - **SPACJA:** Pauza / Wznowienie symulacji.
  - **R:** Reset planszy (wyczyszczenie wszystkich komórek) możliwy tylko podczas pauzy.
  - **ESC:** Wyjście z programu.
  - **Strzałka w Lewo / Prawo:** Zmniejszenie / Zwiększenie prędkości symulacji (FPS).
- **Klawiatura - Wzory i Zapis:**
  - **< (Przecinek) oraz > (Kropka):** Zmiana wybranego wzoru do wstawienia.
  - **S:** Zapisz obecny stan gry do pliku (`zapis_gry.dat`).
  - **L:** Wczytaj stan gry z pliku.

Plik z zapisem gry po kliknięciu klawisza "S" znajduje się w folderze gry. Należy pamiętać, że można wczytać dany plik z zapisem gry klawiszem "L", tylko gdy jest ona otwarta w oknie o tym samym rozmiarze co stan wcześniej zapisanej gry.

6.2 Tryby gry

W menu startowym użytkownik może wybrać tryb **Torus**.

- Jeśli zaznaczony: Plansza jest "nieskończona", co oznacza że obiekty wychodzące z prawej strony pojawiają się z lewej (i analogicznie góra/dół). Zachowuje się ona jakby była "zszyta" krawędziami.
- Jeśli odznaczony: Krawędzie planszy są "ścianami", więc komórki, które tam dotrą, znikają (są usuwane przez mechanizm czyszczenia marginesów).

7 Podsumowanie

Stworzony program to w pełni funkcjonalna implementacja Gry w Życie. Dzięki zastosowaniu operacji macierzowych (`NumPy`), symulacja działa płynnie nawet przy dużych rozmiarach planszy. Dodatkowe funkcjonalności, takie jak edytor planszy w czasie rzeczywistym, biblioteka gotowych wzorów oraz system zapisu stanu, czynią aplikację kompletnym narzędziem do eksperymentowania z automatami komórkowymi.

Warto wspomnieć również, że w programie zaimplementowano mechanizm wizualizacji wieku komórek. Nowo powstałe komórki oznaczane są kolorem jasnozielonym. Wraz z każdym kolejnym przeżytym cyklem symulacji (pokoleniem), ich barwa płynnie zmienia się w kierunku odcieni niebieskiego. Pozwala to na łatwą obserwację dynamiki populacji oraz szybkie odróżnienie struktur stabilnych (starych) od tych, które dopiero powstały.

8 Źródła

- `https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life`
  Ogólny zamysł oraz zasady działania gry.
- `https://playgameoflife.com/`
  Sprawdzenie jak gra działa w praktyce oraz zauważenie przydatnych funkcjonalności.
- `https://playgameoflife.com/lexicon`
  Bardzo duża baza struktur oraz wzorców.
- `https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.convolve2d.html`
  Źródło informacji o funkcji wykonującej splot macierzy.