Dokumentacja “Gra w Życie”

Maciej Słomkowski

Wprowadzenie

Gra w życie (ang. Game of Life) to popularna symulacja komórkowa wymyślona przez matematyka Johna Conwaya w 1970 roku. Pomimo swojej prostoty, gra znajduje wiele życiowych implementacji w dziedzinach nauki

Gra w życie odbywa się na dwuwymiarowej planszy składającej się z komórek. Każda komórka może znajdować się w jednym z dwóch stanów: żywa lub martwa. Stany komórek są aktualizowane zgodnie z określonymi regułami w kolejnych generacjach, co prowadzi do ewolucji układu.

Reguły

Reguły gry w życie są bardzo proste:

1. Każda żywa komórka, która ma mniej niż dwóch żywych sąsiadów, umiera z powodu samotności.
2. Każda żywa komórka, która ma więcej niż trzech żywych sąsiadów, umiera z powodu przeludnienia.
3. Każda żywa komórka, która ma dwóch lub trzech żywych sąsiadów, przetrwa do kolejnej generacji.
4. Każda martwa komórka, która ma dokładnie trzech żywych sąsiadów, staje się żywa w następnej generacji.

Reguły te prowadzą do różnorodnych wzorców, które mogą się pojawiać w trakcie ewolucji gry. Niektóre układy zachowują się stabilnie, inne poruszają się lub tworzą skomplikowane struktury.

Opis projektu

Ten projekt implementuje grę w życie przy użyciu języka Python i biblioteki Pygame. Symulacja odbywa się na dwuwymiarowej planszy, gdzie każda komórka może być w stanie żywym lub martwym. Gracz może interaktywnie ustawiać stany komórek i uruchamiać ewolucję gry.

main.py

Ten plik jest punktem wejścia do aplikacji. Tworzy obiekt klasy `GameOfLife` z pliku `gameOfLife.py` i uruchamia grę.

gameOfLife.py

W tym pliku znajduje się główna logika gry Game of Life. Zawiera on klasę `GameOfLife`, która zawiera metody do manipulowania i wyświetlania stanu gry. Oto krótki przegląd najważniejszych funkcji:

- `__init__`: Konstruktor klasy `GameOfLife`, inicjalizuje podstawowe zmienne gry, takie jak rozmiar siatki, stan gry, prędkość klatek na sekundę (FPS), kolor komórek itp.
- `change_size`: Metoda do zmiany rozmiaru siatki.
- `create_grid`: Metoda do tworzenia pustej siatki gry.
- `draw_grid`: Metoda do rysowania siatki na ekranie.
- `update_grid`: Metoda do aktualizacji siatki na podstawie reguł gry w życie.
- `get_cell_pos`: Metoda do wyznaczania pozycji komórki na podstawie pozycji myszy.
- `start_screen`, `second_screen`, `fps_menu`, `size_menu`, `color_menu` i `menu`: Metody do wyświetlania różnych ekranów/menu gry.
- `run`: Metoda do uruchomienia głównej pętli gry.
- `change_color`: Metoda do zmiany koloru komórek w grze.

Metoda create_grid()

Metoda ta tworzy siatkę o określonym rozmiarze (liczbie wierszy i kolumn), wypełniając ją zerami, które reprezentują "martwe" komórki. Używa ona biblioteki numpy do utworzenia dwuwymiarowej tablicy.

Metoda update_grid()

Ta metoda jest sercem symulacji Gry w Życie. Iteruje przez każdą komórkę w siatce i sprawdza jej sąsiadów. Jeżeli komórka jest żywa i ma mniej niż 2 lub więcej niż 3 sąsiadów, umiera

(zostaje przypisane False). Jeżeli komórka jest martwa i ma dokładnie 3 sąsiadów, staje się żywa (zostaje przypisane True). Wszystkie zmiany są najpierw zapisywane do tymczasowej kopii siatki, a na końcu cała siatka jest zastępowana nową siatką.

Metoda draw_grid()

Metoda ta jest odpowiedzialna za narysowanie siatki na ekranie. Przechodzi przez każdą komórkę w siatce, a jeżeli komórka jest żywa (wartość True), rysuje kwadrat o określonym rozmiarze w odpowiednim miejscu na ekranie.

Metoda menu()

Ta metoda wyświetla menu główne gry. W menu głównym są trzy przyciski: FPS, rozmiar i kolor. Kiedy gracz kliknie jeden z przycisków, jest przenoszony do odpowiedniego podmenu.

Metoda fps_menu()

Metoda ta wyświetla menu FPS, które pozwala graczowi wybrać liczbę klatek na sekundę dla symulacji. Menu to zawiera trzy przyciski: 15 FPS, 30 FPS i 60 FPS. Kiedy gracz kliknie jeden z przycisków, wartość FPS gry jest zmieniana na wybraną wartość i gracz jest przenoszony z powrotem do menu głównego.

button.py

Ten plik zawiera klasę `Button`, która reprezentuje przycisk w interfejsie użytkownika. Posiada metody do rysowania przycisku na ekranie i sprawdzania, czy przycisk został naciśnięty.

W każdym z tych plików mogą być dodatkowe funkcje pomocnicze i klasy, które wspierają główne funkcjonalności. Proszę skonsultować się z komentarzami w kodzie dla szczegółowych informacji na temat działania każdej funkcji i klasy.