LangtonAntor

Program zapisuje obrazy generowane przez mrówkę Langtona na mapie ustalonej przez użytkownika.

Sposób korzystania z interfejsu:

1. Rozpocznij działanie programu poprzez komendę '$python3 terminal_ui.py'.
2. Wybierz kreator mapy.
3. Podaj dane potrzebne do utworzenia mapy.
4. Wybierz folder, do którego zapisane zostaną zdjęcia mapy(ścieżka relatywna).
5. Wybierz, czy chcesz zapisać każde zdjęcie mrówki.
   5.1. Jeśli wybrałeś **tak**, zdjęcia każdego kroku mrówki zostaną wygenerowane w folderze o wcześniej podanej ścieżce.
   5.2. Jeśli wybrałeś **nie**, program przechodzi do "TIMESKIP MODE".
6. Jeśli wszystkie dane podane przez ciebie były prawidłowe, program kończy swoją pracę. W przeciwnym razie program wraca do początku i prosi cię o ponowne wprowadzenie danych.

ad 2. Program obsługuje 3 różne kreatory map:

- 'white' - biała mapa o zadanych wymiarach,
- 'random' - czarno-biała mapa, użytkownik podaje procentową szansę na ukazanie się czarnego pola,
- 'from_image' - użytkownik wybiera zdjęcie, a ono zostaje zamienione na czarno-białą mapę.

ad 3. Zależnie od wybranego kreatora, program będzie wymagał następujących danych:

- 'white' - szerokość, wysokość mapy,
- 'random' - szerokość, wysokość mapy, procentowa szansa na pojawienie się czarnego pola od 0% do 100%,
- 'from_image' - **relatywna** ścieżka do obrazu, który chcemy przekonwertować na mapę.

ad 4. Użytkownik musi podać **relatywną** ścieżkę do folderu, który zostanie utworzony, aby zapisać tam zdjęcia mapy. Jeśli podany folder już istnieje, wszystkie zdjęcia o rozszerzeniu .png zawarte w nim zostaną usunięte, a na ich miejsce zostaną wygenerowane nowe zdjęcia.

ad 5.2 "TIMESKIP MODE" pozwala użytkownikowi na zapisanie stanu mapy po każdym "przeskoku czasowym"
Użytkownik wprowadza ile kroków ma wykonać mrówka pomiędzy każdym "przeskokiem czasowym" oraz ile "przeskoków czasowych" program ma przeprowadzić.

Aspekty techniczne projektu:

Potrzebne biblioteki:

- ❏ numpy
- ❏ Pillow
- ❏ os
- ❏ random
- ❏ *pytest(do testów)*

Projekt podzieliłem na 3 moduły:

1. terminal_ui.py - moduł odpowiedzialny za komunikację z użytkownikiem.
2. map.py - moduł, w którym zdefiniowałem klasę mapy oraz błędy które muszę obsługiwać. W klasie mapy zdefiniowane są metody odpowiedzialne za:
    a. kreatory mapy,
    b. metody odpowiedzialne za tworzenie/oczyszczanie katalogów w których pojawią się zdjęcia map,
    c. metoda zajmująca się przeprowadzaniem mrówki przez wędrówkę, co łączy się z przekazywaniem mrówce koloru pola na którym obecnie się znajduje,
    d. metoda zapisująca zdjęcie ukazujące obecny stan mapy do podanego folderu.
3. ant.py - moduł, w którym zdefiniowałem klasę mrówki. Znajdują się w nim metody odpowiedzialne za:
    a. zwracanie wartości atrybutów mrówki
    b. zmianę atrybutów mrówki(w szczególności zmianę kierunku ruchu)
    c. wzbudzenie mrówki do wykonania kroku oraz zwrócenie kodu koloru na jaki pole powinno się zmienić
    d. sprawdzenie w którą stronę mrówka nie może się poruszyć(_get_illegal_directions)

ad 2.a. Obecna wersja obsługuje 3 kreatory tablic dwuwymiarowych, czyli instancji klasy numpy.ndarray które przechowują kody kolorów w skali grayscale(255 to biały, 0 to czarny). Zależnie od wartości creator_code który podajemy podczas tworzenia instancji klasy mapy, wywoływana jest jedna z metod tworząca instancję tablicy dwuwymiarowej.

ad 2.c. metoda ants_journey pozwala nam zainicjować ruch mrówki po mapie. Metoda pobiera obecne koordynaty mrówki, wysyła mrówce informacje o tym jaki jest kolor pola na którym się znajduje, po czym mrówka wykonuje swój ruch oraz oddaje mapie kod koloru jaki pole mapy ma przybrać. Zmiana argumentu save_every_step na False pozwoliła mi na dziecinnie prostą implementację “TIMESKIP MODE”.

ad 3.a. potrzebowałem dwóch różnych metod do zmiany kierunku mrówki po to, aby móc obsłużyć przypadek w którym mrówka znalazłaby się w sytuacji w której jej ruch przed siebie spowodowałby wyjście poza obręb mapy.

ad 3.c. metoda step()

- ➔ pobiera kolor pola na którym znajduje się mrówka
- ➔ zależnie od tego koloru zmienia kierunek mrówki
- ➔ wzywając metodę _get_illegal_directions otrzymuje listę kierunków w których stronę mrówka nie może się poruszyć
- ➔ jeśli mrówka jest zwrócona w nielegalnym kierunku, to dzięki metodzie _new_direction_random obiera inny losowy kierunek.
- ➔ mrówka porusza się o jedno pole do przodu
- ➔ metoda zwraca kolor na jaki zmienić ma się pole które mrówka opuściła

Problemy napotkane podczas realizacji projektu (oraz ich rozwiązania):

Zdecydowałem się na zapisywanie stanu kratek mapy za pomocą skali grayscale, gdyż już we wczesnej fazie projektowania zauważyłem poważne problemy z tworzeniem zdjęć map gdy zachowywałem wartości kolorów za pomocą wartości True or False. Tworzenie obrazów z tablic z zakodowanymi w nich kolorami w RGB również niosło za sobą problemy z jakością generowanych obrazów(małym plusem tego rozwiązania jest to, iż łatwo mogę modyfikować odcień kolorów na szare).

Problematyczna okazała się kwestia wskazywania na to, które zdjęcie chcemy przekonwertować na mapę oraz do jakiego folderu chcemy zapisać stany wybranej mapy. W ostateczności uznałem że proszenie użytkownika o podanie ścieżki relatywnej do modułu projektu jest akceptowalne.

Refleksja

Pracę nad projektem rozpocząłem od poznania możliwości bibliotek Pillow oraz numpy. Z początku myślałem że program będzie mógł wgrywać tylko czarno białe obrazy, lecz zagłębiając się w zasoby internetu udało mi się wydobyć metody .convert oraz .point które pozwoliły mi na przekonwertowanie dowolnego obrazu na mapę(choć przyznam, iż wyglądają one niepokojąco).

Pomysł z “TIMESKIP MODE” zdawał się naturalnym rozszerzeniem do funkcji które miałem zrealizować, gdyż pozwolił mi w szybki sposób sprawdzić stan obrazu nawet po kilku milionach ruchów mrówki. Zależało mi na tym, gdyż dopiero po kilkunastu tysiącach ruchów możemy zauważyć jak mrówka zaczyna tworzyć tak zwaną “autostradę”. Implementacji tego trybu wskazuje na to w jak prosty sposób możemy poszerzać funkcjonalność programów korzystających z zaproponowanych przeze mnie modułów.

Na koniec mogę dodać, iż realizacja projektu sprawiła mi dużo satysfakcji. Jako że przygodę z programowaniem rozpocząłem niedawno, spodziewam się że można to było zrobić o wiele lepiej.

Mateusz Krakowski