

Paragrafówka

Dla użytkownika

Program stanowi przykład prostej gry tekstowej, w której zadaniem gracza jest przemieszczanie się po pomieszczeniach w celu odnalezienia wyjścia. Program wyświetla opis obecnego pomieszczenia i możliwe akcje, z których gracz może wybierać, wpisując odpowiadające im symbole i zatwierdzając je przyciskiem Enter.

Uruchomienie programu wymaga interpretera Python w wersji co najmniej 3.6.

Windows

Ściągnij i zainstaluj Python. Następnie otwórz plik `main.py` za pomocą Pythona.

Linux

Ściągnij i zainstaluj Python. Pythona możesz znaleźć również w repozytorium swojej dystrybucji, dla przykładu:

- Ubuntu - wpisz w terminalu `sudo apt install python3`
- Arch - wpisz w terminalu `sudo pacman -S python`

Sprawdź, czy posiadasz wystarczającą wersję Pythona, wpisując `python3 -V`.

Aby uruchomić program, wpisz `python3 main.py`.

MacOS

Program nie był testowany na tym systemie, ale powinien działać. Ściągnij i zainstaluj Python, a następnie otwórz plik `main.py`, wpisując `python3 main.py` w terminalu.

Dla programisty

Program wymaga wersji 3.6+ Pythona ze względu na wykorzystanie interpolacji łańcucha znaków.

Projekt podzielony jest na dwa pliki:

- `main.py` - główna pętla i logika programu.
- `classes.py` - klasy opisujące elementy świata gry.

`classes.py`

Klasa `Location`

Przedstawia pojedynczą **lokację** w świecie gry. Posiada krótki opis (`short_desc`) wyświetlany, gdy gracz sąsiaduje z tą lokacją, długi opis (`long_desc`) wyświetlany, gdy gracz znajduje się w tej lokacji oraz opcjonalny opis gdy gracz znajduje się w jednej z wielu części pokoju (`same_room`).

Konstruktor `__init__(self, short_desc, long_desc, same_room = None)`

`short_desc` to łańcuch znaków prezentujący krótki opis lokacji. `long_desc` to łańcuch znaków prezentujący rozbudowany opis lokacji. `same_room` to łańcuch znaków prezentujący krótki opis lokacji gdy gracz jest w części lokacji. Ten opis nie istnieje dla wszystkich lokacji.

Klasa `World`

Przedstawia **świat** gry złożony z lokacji.

Pole `data` przechowuje informację na temat możliwych lokacji w postaci dwuwymiarowej tablicy liczb całkowitych lub znaków. Wartość 0 reprezentuje brak lokacji. Wartości różne od zera są powiązane z obiektami klasy `Location` w sposób określony przez słownik `descriptions`.

Niektóre klucze w słowniku `descriptions` mają specjalne znaczenie:

- 'E' - wyjście/koniec gry

Pola `width` i `height` określają szerokość i wysokość świata gry, w tym przypadku rozmiary tablicy `data`.

Koordynaty świata

Koordynaty świata odpowiadają indeksom używanym z polem `data` obiektu klasy `World`. Pozycja (x, y) w świecie odpowiada wyrażeniu `data[y][x]`. Stąd wynika, że dozwolone wartości x i y są z przedziału odpowiednio `[0, self.width)` i `[0, self.height)`. Wyróżnia się następujące kierunki:

- malejące wartości y - **polnoc**
- rosnące wartości y - **poludnie**
- malejące wartości x - **zachod**
- rosnące wartości x - **wschod**

Konstruktor `__init__(self, mapa)`

Inicjalizuje obiekt. `mapa` nadaje **mapę** światu.

`exist(self, x, y)`

Zwraca wartość typu bool. Mówi, czy na pozycji (x, y) w świecie znajduje się lokacja, do której może udać się gracz. Zakres wartości x, y jest dowolny.

`getAt(self, x, y)`

Zwraca obiekt klasy `Location` znajdujący się na pozycji (x, y). Metoda zakłada, że taki obiekt istnieje. Jego istnienie można sprawdzić za pomocą metody `exist`.

Klasa `Character`

Reprezentuje **gracza** poprzez jego aktualną pozycję w świecie gry. (patrz: koordynaty świata)

Konstruktor `__init__(self, x, y, hp)`

x i y to liczby całkowite określające pozycję gracza. hp jest liczbą całkowitą określającą **punkty Życia** gracza.

main.py

Zmienne globalne

mapp - dwuwymiarowa tablica liczb zawierająca mapę.

world - instancja klasy **World**.

hero - instancja klasy **Character**, **hero.x** oraz **hero.y** są inicjalizowane do pozycji startowej, z której gracz rozpoczyna grę.

ended - zmienna boolowska kontrolująca główną pętlę gry.

endCoordinates - lista zawierająca pozycję końcowej lokacji.

wrongAction - zmienna boolowska pozwalająca wydrukować komunikat gdy gracz poda niewłaściwe polecenie.

keys - słownik wiążący klawisze z odpowiadającymi im nazwami opcji.

Program wykonuje dwie funkcje, służące do ustalenia koordynat końcowej lokacji oraz by dodać do mapy losowe pułapki, następnie drukuje wiadomość rozpoczęcia gry.

Program wtedy rozpoczyna główną pętlę kodu:

```
while not Ended:
    mainLoop()
```

functions.py

addTraps(where)

Dodaje losowo ustawione pułapki do obiektu **where**. Zwraca zmodyfikowane **where.data**.

findEnd(where)

Znajduje lokację końcową w obiekcie **where** i zwraca ją w postaci listy, gdzie pierwszy element to pozycja x, a drugi - pozycja y.

endDirection()

Zwraca łańcuch znaków reprezentujący kierunek, w jakim znajduje się wyjściowa lokacja, w odniesieniu do pozycji gracza.

clearScreen()

Czyści ekran konsoli, wywołując komendę odpowiednią dla danego systemu operacyjnego.

mainLoop()

Zawiera całą interakcję z użytkownikiem.

printCurrentLocation()

Wypisuje długi opis **long_desc** lokacji, w której obecnie znajduje się gracz.

printMenu()

Wypisuje pozycję gracza, kierunek świata w którym znajduje się cel, zdrowie gracza i wszystkie możliwe opcje i kierunki, w jakich gracz może się w danej sytuacji poruszyć. Innymi słowy, prezentuje krótkie opisy **short_desc** lokacji sąsiadujących do tej, w której gracz się obecnie znajduje. Jeśli typ lokacji jest taki sam, jak ten w której znajduje się gracz, to wyświetla **same_room**.

printChoice(x, y, key_name)

Wypisuje wybór w formacie: {nazwa_opcji}: {krótki_opis_lokacji}. x, y to liczby całkowite opisujące pozycję (x, y) lokacji, a **key_name** to łańcuch znaków wskazujący na klawisz powiązany z daną opcją. **nazwa_opcji** jest uzyskiwana dzięki wywołaniu funkcji **getFullOptionName(key_name)**. Jeśli x i y są puste, wypisuje opcje niezwiązane z położeniem.

getFullOptionName(key_name)

Zwraca łańcuch znaków postaci [{klawisz}] {nazwa_opcji}.

checkInput(player_choice)

Sprawdza łańcuch znaków wprowadzony przez gracza i odpowiednio na niego reaguje.

Zwraca kierunek wybrany przez gracza reprezentowany przez dwie liczby całkowite, kolejno **direction_x** (-1 - zachód, 1 - wschód) oraz **direction_y** (-1 - północ, 1 - południe) (patrz: koordynaty świata). Gracz nie może poruszać się na ukos, stąd zawsze jedna z tych zwracanych wartości jest równa 0.

changeRooms()

Zmienia pokój, w którym przebywa gracz, jeśli ta opcja jest możliwa.

moveHero(direction_x, direction_y)

Zmienia pozycję gracza na **next_pos_x = hero.x + direction_x** i **next_pos_y = hero.y + direction_y**, jeśli lokacja na pozycji (**next_pos_x**, **next_pos_y**) istnieje. **direction_x** i **direction_y** to liczby całkowite określające przesunięcie gracza w świecie. (patrz: koordynaty świata)

Jeśli lokacja, do której gracz chce się udać, nie istnieje, zmienna globalna **wrongAction** jest ustawiana na **True**.

Wkład pracy

Bartłomiej Zięba - programowanie, stworzenie dokumentacji.

Kacper Tomasik - programowanie, korekta dokumentacji, fabuła, usprawnianie gry.