Dokumentacja gry w statki

Maciej Ziobro

15 stycznia 2024

1 Wprowadzenie

Kod ten jest implementacją gry w statki. Celem gry jest zatopienie wszytkich okrętów przeciwnika zanim on zrobi to samo. Każdy ruch składa się z wybrania pola lewym przyciskiem myszy. Kliknięcie na pole, jest równozanczne ze strzałem w to pole. Jeśli strzał trafi w okręt, to pole zmieni wygląd. Jeśli strzał nie trafi w okręt, to pole zostanie oznaczone białą kropką. Te same oznaczania obowiązują po stronie przeciwnika (bota), z dodatkiem pola zaznaczonego na czerwono, które oznacza ostatni ruch bota. Dodatkowo okręt całkowicie zatopiony zmienia swój wygląd. Gracz może też prawym przyciskiem myszy oznaczyć dla siebie pola czarną kropką - oznacznie to nie ma żadnego wpływy na rozgrywke.

Każdą rozgrywke gracz znaczyna od wyboru kto wykona pierwszy ruch, istnieje też możliwość wyboru losowego. Po kożdej rozgrywcę gracz może wybrać czy chce zagrać ponownie, czy też zakończyć grę.

2 Plik 1: ppoints.py

2.1 Opis Modułu

Moduł dostarcza klase `Point` reprezentującą punkt na płaszczyźnie.

2.2 Klasa `Point`

Klasa `Point` reprezentuje punkt na płaszczyźnie. Inne części programu korzystają z tej klasy do reprezentacji pól na planszy, pól zajmowanych przez statki, itp. Do inicjalizacji obiektu klasy `Point` potrzebne są współrzędne x i y punktu. Klasa `Point` dostarcza też metod do porównywania punktów, oraz ich tekstowej reprezentacji.

3 Plik 2: ship.py

3.1 Klasa `Ship`

Klasa `Ship` reprezentuje statek. Z obiektu statku możemy otrzymać informacje o jego położeniu, długości, zajmowanych polach, polach dookoła, o tym które jego pola zostały trafione, oraz o tym czy został zniszczony.

3.2 Metoda `__init__`

```
def __init__(self, x1, y1, orientation, length):
```

Inicjalizuje obiekt klasy Ship.

Argumenty

- `x1 (int)`: Współrzędna x lewego górnego punktu statku.
- `y1 (int)`: Współrzędna y lewego górnego punktu statku.
- `orientation (bool)`: Orientacja statku: True dla położenia poziomego, False dla pionowego.
- `length (int)`: Długość statku.

Zwraca

None

3.3 Metoda `__str__`

Zwraca

- `str`: Reprezentacja tekstowa obiektu klasy Ship.

3.4 Właściwość `is_destroyed`

Sprawdza, czy statek został zniszczony poprzez sprawdzenie, czy wszystkie jego punkty zostały trafione.

Zwraca

- `bool`: True, jeśli statek został zniszczony, False w przeciwnym razie.

3.5 Właściwość `is_destroyed_draw`

Sprawdza, czy statek został zniszczony. Jest to kopia metody `is_destroyed`, która jest używana tylko do wyświetlania statków.

Zwraca

- `bool`: True, jeśli statek został zniszczony, False w przeciwnym razie.

3.6 Metoda `list_of_points`

Zwraca

- `list`: Lista obiektów Point, przedstawiających punkty zajmowane przez statek.

3.7 Metoda `extend_list_of_points`

Zwraca

- `set`: Zbiór obiektów Point, przedstawiających punkty zajmowane przez statek i pola sąsiednie.

3.8 Metoda `strike`

Oznacza punkt statku jak trafiony.

Argumenty

- `point (Point)`: Punkt do oznaczenia jako trafiony.

3.9 Metoda `strike_draw`

Oznacza punkt jako trafiony przez statek. Jest kopią metody `strike` natomiast słóży do wyświetlania statków.

Argumenty

- `point (Point)`: Punkt do oznaczenia jako trafiony.

4 Plik 3: bot.py

4.1 Opis Modułu

Moduł ma na celu symulowanie bota, będącego przeciwnikiem gracza. Bot wykonuje ruchy w sposób losowy, jednak po trafieniu w statek dostosowuje swoją strategię do sytuacji. Główna funckja modułu jest `bot_move`, a wszytkie pozostałe funckje są wywoływane przez nią.

4.2 Funkcja `bot_move`

Jest to główna funckja modułu, która wykonuje ruchy bota. Jeżeli bot nie trafił w żaden statek to wywołując funkcję `new_move`, wybiera losowe pole. Jeżeli bot trafił w statek, to wywołuje funkcję `after_strike_move`, która wybiera kolejne pola do strzału. Tak długo jak bot nie zestrzeli wszystkich statków przeciwnika, będzie wykonywał ruchy w sposób opisany powyżej, i dodawał je do listy ruchów.

Argumenty

- `ships_placment (list)`: Lista obiektów klasy Ship, przedstawiająca statki gracza.

Zwraca

- `list_of_moves (list)`: Lista obiektów klasy Point, przedstawiająca ruchy kolejne bota.

4.3 Funkcja `new_move`

Funkcja wybiera losowe pole, następnie sprawdza czy pole to nie zostało już użyte (bot ma informacje, że na tym polu na pewno nie ma niezatopionego statku). Jeżeli pole nie zostało użyte, to jest zwracane jako ruch bota. W przeciwnym przypadku losowanie jest powtarzane. Dodatkowym sprawdzeniem jest to, wykonywane przez funkcję `ship_space` - opisną w dalszej części dokumentacji.

Argumenty

- `ships_placment (list)`: Lista obiektów klasy Ship, przedstawiająca statki gracza.
- `points_used (list)`: Lista obiektów klasy Point, mówi o punktach użytych przez bota lub takich o których wiadomo, że nie ma tam statku.
- `current_ship (list)`: Lista obiektów klasy Point, zawiera punkty statku, który został trafiony ale nie zatopiony - jest aktualnie atakowany przez bota.

Zwraca

- `point (Point)`: Obiekt klasy Point, oznaczający następny ruch bota.

4.4 Funkcja `after_strike_move`

Funkcja na początek sprawdza, czy znany jest tylko jeden punkt statku, który został trafiony. Jeżeli tak, to funckja losuje jedno pole z listy pól dookoła tego punktu. Następnie sprawdza czy pole to nie zostało już użyte (bot ma informacje, że na tym polu na pewno nie ma niezatopionego statku). Jeżeli pole nie zostało użyte, to jest zwracane jako ruch bota. W przeciwnym przypadku losowanie jest powtarzane. Jeżeli nastomiast znane są dwa lub więcej punkty statku, które zostały trafione, to funckja losuje jedno pole z listy pól dookoła tych punktów z uwzględnieniem tego w jakim kierunku położony jest statek. Następnie sprawdza czy pole to nie zostało już użyte. Dodatkowym sprawdzeniem jest to, wykonywane przez funkcję `ship_space` - opisną w dalszej części dokumentacji.

Argumenty

- `ships_placment (list)`: Lista obiektów klasy Ship, przedstawiająca statki gracza.
- `points_used (list)`: Lista obiektów klasy Point, mówi o punktach użytych przez bota lub takich o których wiadomo, że nie ma tam statku.
- `current_ship (list)`: Lista obiektów klasy Point, zawiera punkty statku, który został trafiony ale nie zatopiony - jest aktualnie atakowany przez bota.

Zwraca

- `point (Point)`: Punkt oznaczający następny ruch bota.

4.5 Funkcja `ship_space`

Funkcja sprawdza czy w danym kierunku jest dostępna przestrzeń dla najmniejszego niezatopionego statku. **Argumenty**

- `point (Point)`: Lista obiektów klasy Point, zawiera punkty statku, który został trafiony ale nie zatopiony - jest aktualnie atakowany przez bota.
- `points_used (list)`: Lista obiektów klasy Point, mówi o punktach użytych przez bota lub takich o których wiadomo, że nie ma tam statku.
- `ships_placment (list)`: Lista obiektów klasy Ship, przedstawiająca statki gracza.

Zwraca

- `space (list)`: Lista dwóch wartości bool, mówiąca o tym czy w danym kierunku jest dostępna przestrzeń dla najmniejszego niezatopionego statku.

5 Plik 4: main.py

5.1 Opis Modułu

Moduł zawiera główną część programu. Zawiera cały kod odpowiedzialny za GUI gry.

5.2 Funkcja `ships_placement`

Funkcja generuje rozmieszczenie statków na planszy gry. Na początku tworzy zbiór punktów z których będą losowane punkty początkowe statków. Następnie dla każdego rozmiaru statku z listy, losuje statki w odpowiedniej ilości. W trakcie procesu losowania sprawdzane jest czy statek może zostać umieszczony na planszy. Jeśli nie, to całe losowanie jest powtarzane. Dodatkowo w sytuacji, w której nie można umieścić już żadnego statku, na planszy, funkcja jest wywoływana ponownie.

Argumenty

- `list_of_sizes (list)`: Lista ilości statków poszczególnego rozmiaru od największego do najmniejszego. Zgodnie z standardowymi regułami gry w statki, mamy 1 statek o długości 4, 2 statki o długości 3, 3 statki o długości 2 i 4 statki o długości 1.

Zwraca

- `list_of_ships (list)`: Lista obiektów klasy Ship, reprezentujących rozmieszczenie statków.

5.3 GUI

Druga część modułu zaczyna się od pętli `while`, która jest odpowiedzialna za zresetowanie gry gdyby gracz chciał zagrać ponownie. W pętli kolejno znajdują się:

1. Ustawienia początkowe FPS

2. Utworzenie listy statków gracza i bota poprzez wywołanie funkcji `ships_placement`
3. Utworzenie listy ruchów bota poprzez wywołanie funkcji `bot_move` z modułu `bot.py`
4. Utworzenie list obiektów `pygame.Rect` reprezentujących pola planszy gracza i bota
5. Definicje stały związanych z wyświetalniem elementów na ekranie
6. Definicje czcionek
7. Funkcja `start_menu`, odpowiedzialną za wyświetlenie menu startowego
    - Funkcja pozwala na wybór kto wykona pierwszy ruch, co rozpoczyna rozgrywke
8. Funckja `end_menu`, odpowiedzialną za wyświetlenie menu końcowego
    - Funkcja pozwala na wybór czy gracz chce zagrać ponownie, czy wyjść z gry
9. Inicjalizacja obrazków, które potem będą użyte do wyświetlania pól
10. Zmienne pomocnicze - opisane komentarzami w kodzie
11. Pętla `while`, która jest główną pętlą gry
    (a) Pętla `for`, która obsługuje eventy `pygame.event`

    Pętla obsługuje kliknięcia myszy, oraz wyjście z gry.
        - Lewy przycisk myszy - strzał gracza
        - Prawy przycisk myszy - oznaczenie pola (nie ma wpływu na rozgrywke)

    (b) Sprawdzenie czy gracz wygrał

    (c) Wykonanie ruchu bota

    (d) Sprawdzenie czy bot wygrał

    (e) Dodanie napisów na ekranie
        - Podpisy pól

- Napis informujący o tym ile danych statków dana strona gry ma, i czy są zatopione
- Napis informujący czyja tura nastepuje

(f) Seria pętali `for`, odpowidzialnych za wyświetlenie pól

- Wyświetlenie wszytkich pól z tłem morza
- Zaznacznie wszytkich pól, które zostały użyte przez zarówno gracza jak i bota - biała kropka
- Rysowanie statków z uwzględnieniem rozróżnienia na zniszczone, trafione i nietrafione
- Zaznacznie pól oznaczonych przez gracza czarną kropką
- Zaznacznie ostatniego ruchu bota czerwoną kropką

(g) Wyświetlenie napisów końcowych oraz wywołanie funkcji `end_menu` w przypadku wygranej jednej ze stron