

Dokumentacja gry „Duck hunt!”

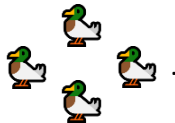




Gra „Duck hunt!” polega na strzelaniu do kaczek za pomocą przypisanych klawiszy dla każdego z graczy. Celem gry jest postrzelenie pięciu kaczek zanim zrobi to przeciwnik. Gracz z najwyższym wynikiem wygrywa grę.

Wymagania wstępne

- Python 3
- Moduł *msvcrt* (tylko dla systemu Windows)
- Moduł *curses*

Zasady gry

- Każdy gracz kontroluje swoją broń za pomocą przypisanych im klawiszy
 - dla gracza pierwszego (Player 1) są to klawisze WASD
 - dla gracza drugiego (Player 2) są to klawisze IJKL.
- Kaczki pojawiają się na ekranie losowo, prawo/lewo/góra/dół .
- Gdy kaczka pojawi się na ekranie, gracze muszą nacisnąć odpowiedni klawisz, aby ją zestrzelić. Miejsce pojawienia się kaczki odpowiada jednemu z czterech klawiszy dostępnych dla każdego z graczy.
- Każde trafienie zwiększa wynik gracza o jeden punkt .
- Gra kończy się, gdy jeden z graczy osiągnie wynik 5 .
- Grę wygrywa gracz z najwyższym wynikiem.

Przegląd kodu

Importy

```
1 import threading
2 import random
3 import msvcrt
4 import curses
```

Zaimportowane zostały niezbędne moduły do obsługi wątków, generowania losowych liczb, obsługi klawiatury i wyświetlania na konsoli.

Zmienne i stałe

```
6 coords = [[6, 29], [10, 20], [14, 29], [10, 39], [6, 89], [10, 80], [14, 89], [10, 99]]
7 keys = ['w', 'a', 's', 'd', 'i', 'j', 'k', 'l']
```

Lista **coords** przechowuje współrzędne pojawienia się kaczek na ekranie. Lista **keys** przechowuje odpowiednie klawisze dla każdego z graczy.

Definicje funkcji

```
9 def getch():
10     return msvcrt.getch().decode('utf-8')
```

Funkcja **getch()** służy do odczytywania wejścia z klawiatury od gracza. Wykorzystuje moduł **msvcrt** do odczytu pojedynczego znaku i dekoduje go do postaci ciągu znaków w formacie UTF-8.

```
12 def generate_key1(screen):
13     key = random.randint(0, 3)
14     screen.addstr(coords[key][0], coords[key][1], "🐣")
15     screen.refresh()
16     return key
17
18 def generate_key2(screen):
19     key = random.randint(4, 7)
20     screen.addstr(coords[key][0], coords[key][1], "🐣")
21     screen.refresh()
22     return key
```

Funkcje **generate_key1()** i **generate_key2()** są odpowiedzialne za generowanie losowych pojawień się kaczek dla gracza 1 i gracza 2. Funkcje wybierają losową współrzędną z listy **coords** i wyświetlają na ekranie emoji kaczki za pomocą **screen.addstr()**. Następnie zwracany jest indeks klawisza do dalszego przetwarzania.

Klasa Game

```

24 class Game:
25     def __init__(self):
26         self.player1_score = 0
27         self.player2_score = 0
28         self.game_over_event = threading.Event()
29
30     def start(self):
31         screen = curses.initscr()
32         num_rows, num_cols = screen.getmaxyx()
33
34         screen.addstr(0, 55, "Duck hunt! 🦆🎯")
35         screen.addstr(10, 20, "A      Player 1      D")
36         screen.addstr(10, 80, "J      Player 2      L")
37         screen.addstr(6, 29, "W")
38         screen.addstr(14, 29, "S")
39         screen.addstr(6, 89, "I")
40         screen.addstr(14, 89, "K")
41         screen.addstr(3, 20, "Score: ")
42         screen.addstr(3, 80, "Score: ")
43
44         screen.refresh()
45
46         thread1 = threading.Thread(target=self.play, args=(1, self.game_over_event, screen))
47         thread2 = threading.Thread(target=self.play, args=(2, self.game_over_event, screen))
48         thread1.start()
49         thread2.start()

```

```

51     def play(self, player_number, game_over_event, screen):
52         score1 = 27
53         score2 = 87
54         while not game_over_event.is_set():
55             if player_number == 1:
56                 key = generate_key1(screen)
57             else:
58                 key = generate_key2(screen)
59             while (True and (not game_over_event.is_set())):
60                 if getch() == keys[key]:
61                     if player_number == 1:
62                         self.player1_score += 1
63                         screen.addstr(3, score1, "🦆")
64                         score1 += 2
65                         screen.addstr(coords[key][0], coords[key][1], (keys[key]).upper() + " ")
66                         screen.refresh()
67                     if self.player1_score == 5:
68                         screen.clear()
69                         game_over_event.set()

```

```

70             else:
71                 self.player2_score += 1
72                 screen.addstr(3, score2, "🦆")
73                 score2 += 2
74                 screen.addstr(coords[key][0], coords[key][1], (keys[key]).upper() + " ")
75                 screen.refresh()
76                 if self.player2_score == 5:
77                     screen.clear()
78                     game_over_event.set()
79             break
80         if self.player1_score == 5:
81             screen.addstr(15, 50, "🏆 Player 1 won! 🏆")
82         else:
83             screen.addstr(15, 50, "🏆 Player 2 won! 🏆")
84         screen.refresh()
85         curses.napms(8000)
86         curses.endwin()

```

Klasa **Game** reprezentuje grę i zawiera metody związane z jej przebiegiem. Konstruktor `__init__()` inicjalizuje początkowe wartości wyników graczy oraz tworzy obiekt Event z modułu **threading** do sygnalizowania zakończenia gry.

Metoda **start()** rozpoczyna grę. Inicjalizowany jest ekran za pomocą `curses.initscr()`, a następnie wyświetlane są tytuł gry i mapowanie klawiszy dla graczy. Wątki są uruchamiane dla obu graczy, z wywołaniem metody **play()**.

Metoda **play()** obsługuje przebieg gry dla konkretnego gracza. Na początku wybierany jest klucz (indeks) dla generowania kaczek zależnie od numeru gracza. Następnie w pętli oczekuje na naciśnięcie klawisza przez gracza. Jeśli naciśnięty klawisz jest zgodny z oczekiwanym kluczem, to odpowiednio zwiększany jest wynik gracza, wyświetlana jest ikona trafienia na ekranie, aktualizowany jest wynik na ekranie oraz sprawdzane jest, czy któryś z graczy osiągnął wynik równy 5. W przypadku osiągnięcia wyniku 5 przez któregoś z graczy, ekran jest czyszczony, a zdarzenie sygnalizujące zakończenie gry jest ustawiane (`game_over_event.set()`).

Po zakończeniu gry, na podstawie wyników graczy wyświetlany jest odpowiedni komunikat o zwycięzcy. Następnie ekran jest odświeżany, opóźnienie 8 sekund jest wprowadzane za pomocą `curses.napms(8000)`, a na koniec ekran jest zamykany (`curses.endwin()`).

Inicjalizacja i uruchomienie gry

```
87 game = Game()
88 game.start()
```

Tworzony jest obiekt gry **Game** i rozpoczynana jest gra za pomocą metody **start()**.