Snake

## Uruchomienie lokalne

Wymagany jest zainstalowany python.

1. Otwórz terminal, przejdź do katalogu z projektem

2. Zainstaluj wymaganie pakiety, wykonując następującą komendę

*pip install -r .\server\requirements.txt*

3. Uruchom serwer

*python server\server.py*

4. Otwórz plik *index.html* w przeglądarce.

## Serwer – backend

W skład serwera wchodzą 3 moduły:

* constants.py
* game.py
* server.py

W pliku *constants.py* znajdują się wydzielone stałe dla zachowania czystości i przejrzystości kodu.

Moduł *game.py* odpowiada za mechanikę gry. Zdefiniowane są w nim dwie klasy: klasa Game – reprezentująca stan gry, oraz klasa Player – która opisuje stan gracza (dokładniej kontrolowanego przez niego węża).  
  
class Game:  
 def \_\_init\_\_(self):  
 self.board\_width = 10  
 self.board\_height = 10  
 self.tick = 1  
 self.players = {}  
 self.started = False  
 self.food = []  
 self.state = State.WAITING\_FOR\_ANOTHER\_PLAYER

board\_width, board\_height – rozmiar planszy  
tick – szybkość gry (jak często stan gry jest aktualizowany i wysyłany do graczy) w sekundach  
players - słownik przechowujący graczy należących do danej sesji gry  
started – flaga określająca czy gra rozpoczęła się  
food – tablica zawierająca koordynaty jedzenia  
state – status gry

class Player:  
 def \_\_init\_\_(self, name, body\_segments, direction, color):  
 self.name = name  
 self.body\_segments = body\_segments  
 self.direction = direction  
 self.last\_direction = direction  
 self.color = color

name – nazwa gracza

body\_segments – tablica koordynatów tworzących danego węża

direction – obecny kierunek poruszania się węża (lewo/prawo/góra/dół)

last\_direction – kierunek w jakim poruszył się wąż w poprzednim ticku  
color – kolor węża

Moduł *server.py* jest na szczycie backendu – to w nim następuje komunikacja z klientami. Komunikacja realizowana jest z użyciem protokołu WebSocket, który zapewnia dwukierunkowy kanał wymiany danych przez pojedyncze połączenie TCP. Umożliwia to przesyłanie danych przy niższym obciążeniu z i do serwera w czasie rzeczywistym (w porównaniu do innych rozwiązań półdupleksowych takich jak np. odpytywanie HTTP).

Asynchroniczne wywołania funkcji realizowane są przy pomocy biblioteki asyncio – z użyciem składni async, await.

Komunikaty wysyłane z i do serwera są w formacie JSON.  
Przykłady komunikatów:

1. Rozpoczęcie sesji (wysyła klient do serwera)

{

"action": "start",

"player": {

"name": "player770730",

"color": "#00F"

}

}

2. Dołączenie do istniejącej sesji (wysyła klient do serwera)

{

"action": "join",

"key": "rS-dlvLqdmy7hkZN",

"player": {

"name": "player142344",

"color": "#F00"

}

}

3. Informacja o sesji (wysyła serwer do klienta)

{

"type": "game\_info",

"width": 10,

"height": 10,

"key": "rS-dlvLqdmy7hkZN"

}

4. Informacja o stanie w trakcie gry (wysyła serwer do klienta)

{

"type": "status",

"food": [

[7, 1]

],

"players": [

{

"name": "player798302",

"bodySegments": [

[1, 6]

],

"color": "#00F"

},

{

"name": "player142344",

"bodySegments": [

[7, 4]

],

"color": "#F00"

}

],

"state": "GAME\_IN\_PROGRESS"

}

5. Informacja o zmianie kierunku węża (wysyła klient do serwera)

{

"action": "move",

"direction": "down",

"playerName": "player142344"

}