15. **Gra Snake**

- Klienci - łączą się z serwerem, czekają na drugą osobę, grają razem, ...
- Serwer - serwer zarządza grami klientów, w danej chwili może być prowadzonych kilka gier, ...

Uwagi

- Zarówno klient jak i serwer, muszą określić protokół przesyłu danych
- Serwer przez cały czas swojego działania zapisuje do logów informacje o aktualnym zdarzeniu (połączył się klient - z jakim IP, portem, ...)
- Serwer i klient powinni obsługiwać protokół IPv4 oraz IPv6
- ...
- **Realizacja:** ...

Wiele gier na raz
jedna gra - dwóch uczestników
Do komunikacji na warstwie transportowej będziemy korzystać z UDP

d1, d2 (direction) - kierunek ruchu węży
stan gry
request_player1: {
	“name”: “game_name”,
	“d1”: “u”,
}

request_player2: {
	“name”: “game_name”,
	“d2”: “l”,
}

JSON
game_state = {
	“name”: “game_name”,
	“d1”: “u”,
	“d2”: “d”,
	“f”: [x, y],
	“pt”: [pt1, pt2],
	“p1_game_over”: 0,
	“p2_game_over”: 0,
	“p1”: [[x, y], [x, y], [x, y]],
	“p2”: [[x, y], [x, y], [x, y]],
}
lub format binarny

Serwer:

- serwer przechowuje aktualny stan gry
- serwer przechowuje aktualny kierunek ruchu węży
- posiada wewnętrzny zegar, który co interwał czasu aktualizuje klientom stan planszy
- przed odesłaniem stanu gry przesuwa węże zgodnie z aktualnym kierunkiem
- przed odesłaniem stanu gry sprawdza kolizje

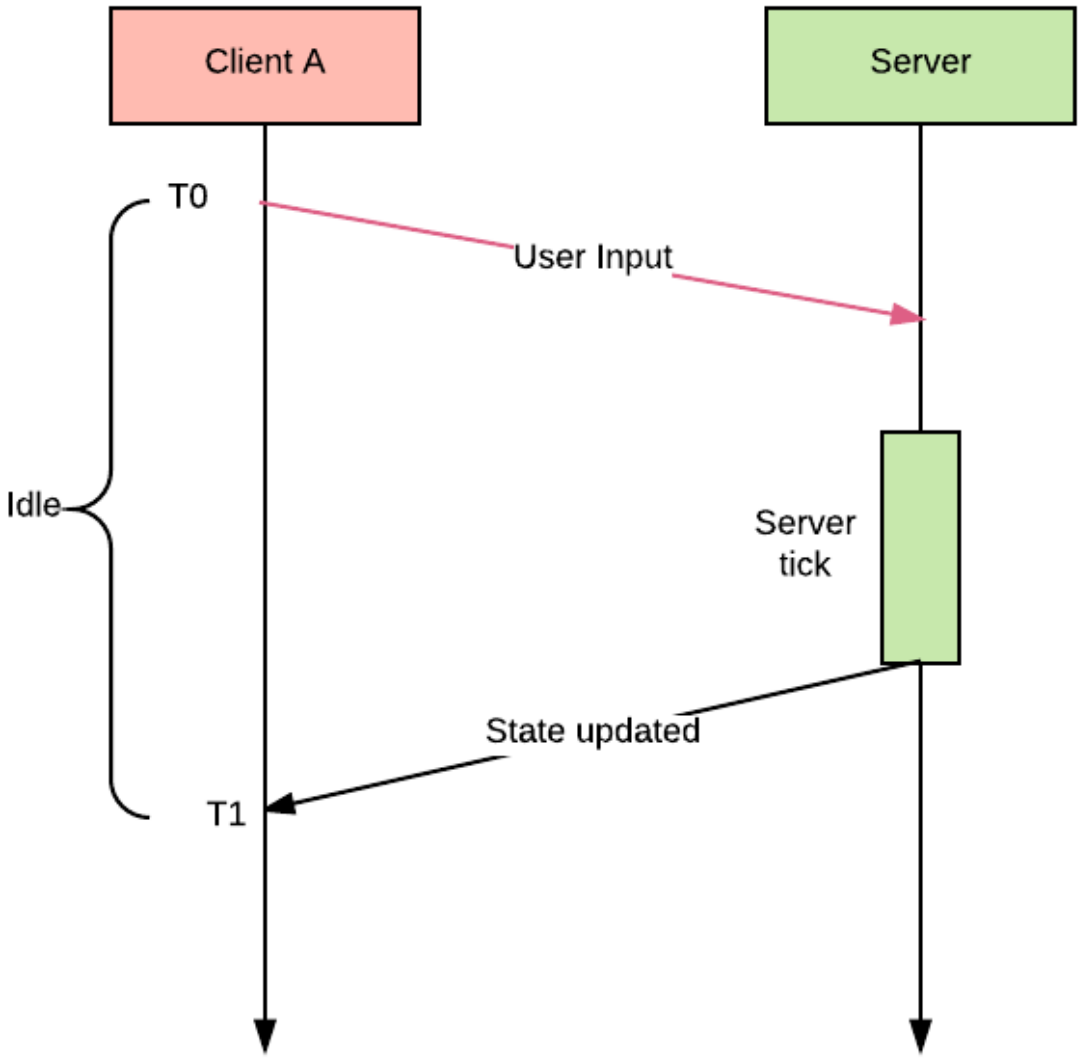
- przed odesłaniem sprawdza jedzenie, wydłuża węże i generuje nowe położenie jedzenia
-

Klient:

while True:

1. Odczytaj kierunek ruchu
2. Wyślij stan gry
3. Odbierz stan gry
4. Rysuj snake (jeśli otrzymano nowy stan gry)

Linki:
https://gist.github.com/PlainSight/5f324f9a215e72fb2539454c88ded5bc
https://github.com/simonwittber/netwrok-server/tree/master/src/netwrok
https://mortoray.com/2020/12/06/high-throughput-game-message-server-with-python-websockets/



Podział obowiązków (MVP) do 1-06-2021:
Bartek - klient, wersja graficzna tkinker, apka desktopowa
Norbert - wystawienie serwera zorientować się w temacie, komunikacja, protokół
Antek - serwer podstawowa obsługa 1 gra, dla 2 graczy

BRUDNOPIS

---

20 maja 2021r.

Synchronizacja multiplayerowa - serwer ma swój czas, klient nie wykona ruchu dopóki serwer nie odpowie mu ze zaktualizowanym stanem, który wysłał przed chwilą klient.
Serwer przechowuje informacje o ostatnich ruchach klientów, co jakiś czas informacje do serwera o ruchach.
Serwer nie powinien przechowywać danych ruchów tylko stan całej gry - położenia i punkty graczy.

Z czego będziemy korzystać (3 dni)?
*** Ja propunuję GitHuba.*
Sphinx (automatyczne dokumentowanie kodu).

---

24 maja 2021r.

Więcej klient będzie wysyłał do serwera.
Na początek dla ułatwienia można by zrobić grę na dwie osoby.
Stan gry - pozycje snake'a, kolejnych części, współrzędne, drugiego snake'a i jedzenia.
Za to, czy jedzenie zostało zjedzone będzie odpowiadał serwer.
Jeżeli któryś z węży znajdzie się w pozycji, gdzie jest jedzenie, serwer automatycznie generuje nową pozycję i przesyła do klientów.
Gra nie "zawija się" - spotkanie ze ścianą kończy grę.

Zarys architektury (kodu), schemat klienta i serwera na czwartek.

Biblioteka async, standardowa w Pythonie - ogarnąć do napisania po stronie serwerowej.

---

27 maja 2021r.

Jeżeli snake znajdzie się w miejscu jedzenia, to serwer powinien w jakiś sposób odesłać informacje o tym fakcie.
U obydwu klientów równocześnie ma mieć miejsce odświeżanie stanu gry.
W czasie "ticku" klienci wysyłają swoje kierunki, a serwer odsyła już stan gry w kolejnym kroku.
Klient nie ma żadnego zegara tylko cały czas czeka na odpowiedzi serwera, które (w pewnym sensie) będą synchronizować grę (serwer będzie wysyłał informacje w określonych odstępach czasu)
Dodatkowe ściany, poziomy trudności (prędkość, itp.).
Serwer dba o poprawność gry - wszystkie operacje dzieją się tak naprawdę na nim.
Klienci jedynie wysyłają swoje ruchy i odbierają stan planszy, który mają wyświetlić.

Przesyłanie danych będzie realizowane poprzez JSON lub podobną odmianę (lub jednak własna enkapsulacja danych).
Komunikacja w UDP.

Dodatkowe rozszerzenia:
* poziomy trudności
* dodatkowe ściany
* zmienna prędkość
* sprawdzanie (po 5-10 sekundach braku danych), czy połączenie zostało utracone
* dodatkowe informacje dla klientów

Spotkanie 09.06.2021
Podsumowanie:
Antek - do piątku ogarnąć MVP, żeby było co pokazać na zajęciach
Norbert - najlepiej do soboty max niedzieli wrzucić protokół+bibliotekę

Antek+Bartek - do wtorku dostosować klienta i serwer do protokołu Norberta
Antek+Norbert - do wtorku ogarnąć działanie na async

Wtorek spotkanie, finalny podział zadań.

CO	KTO
Obsługa ipv4 i ipv6	
zbieranie logów po stronie serwera	
obsługa błędów (serwer)	Norbert
przejście na TCP	Antek
Ulepszenie dokumentacji (Co wyróżnia rozwiązanie)	Bartek
Szyfrowanie TLS	Antek

Co wyróżnia rozwiązanie
Wymagania co do wersji
Biblioteki
z czego jesteśmy najbardziej zadowoleni
Podsumowanie