Kacper Grzelakowski 197897
Jędrzej Jasiniecki 197993
Jakub Stachowicz 198302
Maciej Winiecki 198363

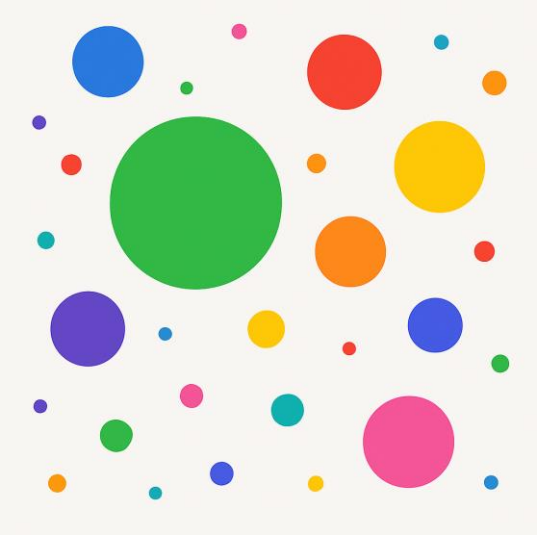
Cellfight

Część 1 i 2

Część 1 – Opis:

Opis gry

CellFight to wieloosobowa gra czasu rzeczywistego, w której gracze sterują komórką poruszającą się po dwuwymiarowej planszy. Celem gry jest pochłanianie mniejszych komórek, zarówno bezwładnych (pokarmu), jak i innych graczy, aby zwiększyć swoją masę. Rozgrywka ma charakter rywalizacyjny i dynamiczny – większe komórki mogą pochłaniać mniejsze, ale stają się przy tym wolniejsze, co otwiera pole do taktyki i unikania zagrożenia. Oprócz przeciwników, na mapie znajdują się również przeszkody i ulepszenia, które wpływają na rozgrywkę.



Zasady gry

- Gracz rozpoczyna grę jako niewielka komórka i porusza się po mapie, zbierając mniejsze cząsteczki pokarmu, co pozwala jego komórce rosnąć
- Komórki większe mogą pochłaniać mniejsze, pod warunkiem, że różnica rozmiarów jest wystarczająca
- Po pochłonięciu innego gracza, masa zwycięzcy zwiększa się
- Gracz może również wyrzucać masę, by przyciągnąć inne komórki lub nakarmić sojuszników
- Na mapie rozmieszczone są przeszkody, które zmniejszają masę gracza lub czasowo go opóźniają
- Na mapie rozmieszczone są ulepszenia, które czasowo zwiększają szybkość gracza lub przydzielają mu tarczę chroniącą przed zagrożeniami
- Rozgrywka nie ma ustalonego końca – gra trwa do momentu pochłonięcia przez innego gracza lub wyjścia z sesji

Sterowanie

Sterowanie w **CellFight** jest intuicyjne i odbywa się za pomocą myszy oraz klawiatury. Kierunek ruchu komórki określany jest przez pozycję kursora myszy – komórka nieustannie przemieszcza się w jego stronę. Dodatkowo naciśnięcie klawisza **W** umożliwia wyrzucenie części masy w kierunku kursora, co może służyć jako taktyczne zagranie (np. karmienie lub wabienie przeciwników).

Wybór architektury

Wybrana została architektura klient-serwer. Przy tym typie gry jest to wybór naturalny – za postaciami w grze stoją klienci, a za planszą – serwer. Jednak sam argument o naturalności wyboru nie jest jedynym. Gra ta, będąca grą rywalizacyjną, musi posiadać elementy rozstrzygania sporów. W przypadku architektury peer-to-peer trudno jest zaprojektować sprawiedliwy mechanizm oceny sytuacji. W dodatku – kto będzie zarządzał pojawianiem się nowych kulek pokarmu? Mechanizm serwera centralnego sam się narzuca. Kolejnym argumentem jest wydajność. Przy większej liczbie klientów architektura peer-to-peer nie będzie wydajna, gdyż trzeba będzie zarówno wysyłać dane do wszystkich innych graczy, jak i od nich te dane odbierać. Pośrednictwo serwera (który powinien być mocniejszą maszyną) powoduje, że poszczególni klienci są odciążeni obliczeniowo, a grać można na słabszym sprzęcie i ze słabszym połączeniem sieciowym.

Synchronizacja danych w architekturze rozproszonej

W kontekście implementacji rozproszonej, gra **CellFight** wymaga odpowiednio zaplanowanych punktów synchronizacji danych pomiędzy klientami i serwerem. Główne elementy synchronizacji to:

- **Pozycja i masa komórek graczy**: te dane są wysyłane od klienta do serwera w stałych odstępach czasu, a następnie serwer przesyła zaktualizowany stan gry do wszystkich graczy, aby utrzymać spójny obraz planszy.
- **Wyrzucenie masy:** wysyłane natychmiastowo jako osobne pakiety zdarzeń (eventy), by zapewnić niskie opóźnienie reakcji.
- **Pojawianie się i znikanie elementów gry** (pokarm, nowe komórki): synchronizowane cyklicznie przez serwer, który zarządza globalnym stanem świata gry.
- **Kolizje i pochłanianie**: obliczane na poziomie serwera, aby uniknąć niespójności wynikających z rozbieżności czasowych między klientami.

Kto i jak decyduje o sprawach spornych?

W środowisku rozproszonym odpowiedzialność za rozstrzyganie spraw spornych, takich jak kolizje, pochłanianie komórek czy wyrzut masy, spoczywa na **serwerze gry**, pełniącym funkcję zaufanego autorytetu. Serwer centralny posiada pełną i aktualną wiedzę o stanie gry, dlatego to

on ostatecznie podejmuje decyzje w sytuacjach niejednoznacznych lub konfliktowych. Przykładowo, w momencie zetknięcia się dwóch komórek, serwer dokonuje obliczeń na podstawie ich pozycji i masy, aby określić, czy dochodzi do pochłonięcia. Klienci przesyłają jedynie informacje o swoich zamiarach (np. kierunku ruchu), natomiast ich rzeczywisty efekt jest symulowany i zatwierdzany przez serwer. Takie podejście zapewnia spójność rozgrywki oraz zabezpiecza przed ewentualnymi oszustwami ze strony klientów.

Wybór protokołu

Protokołem wybranym do komunikacji między serwerem a użytkownikami jest UDP. Zapewnia on szybszy czas działania, lecz nie zapewnia docieralności i kolejności pakietów. Nie są to jednak cechy koniecznie wymagane przez gry sieciowe. Stan gry aktualizowany jest kilkadziesiąt razy na sekundę. Utrata jednego pakietu nie stanowi problemu, podobnie jak otrzymanie pakietu w złej kolejności. Pakiety spoza kolejności zostaną zignorowane, a przy pakietach utraconych aktualizacje odbędą się na zasadzie interpolacji.

Problem pojawi się, kiedy pakietów utraconych będzie dużo w relatywnie długim (jak na działanie gry czasu rzeczywistego) czasie – np. w ciągu kilku sekund. Wówczas należy uznać, że gracz się rozłączył, a jego postać powinna zostać usunięta z gry. Interpolacja zamiarów gracza w przeciągu ciągu kilku sekund jest już niemożliwa.

Stan gry

Stan gry przechowywany i aktualizowany jest na serwerze. Gra ta, choć jest grą czasu rzeczywistego, traci na prędkości rozgrywki wraz ze wzrostem masy gracza. Ułatwia to centralizację obliczeń, a opóźnienia wynikające z komunikacji pomiędzy graczami i aktualizacji na serwerze nie są aż tak widoczne.

Sekcja krytyczna

Choć aktualizacje położeń graczy mogą się odbywać współbieżnie, decyzje kluczowe, głównie te o kolizjach (zwłaszcza tych mogących doprowadzić do pochłonięcia jednego gracza przez innego) powinny mieć zapewnioną ochronę przed wyścigami (ang. race condition). Taką ochronę zapewnia podejście jednowątkowe. W związku z tym, sekcją krytyczną zajmie się tylko jeden wątek.

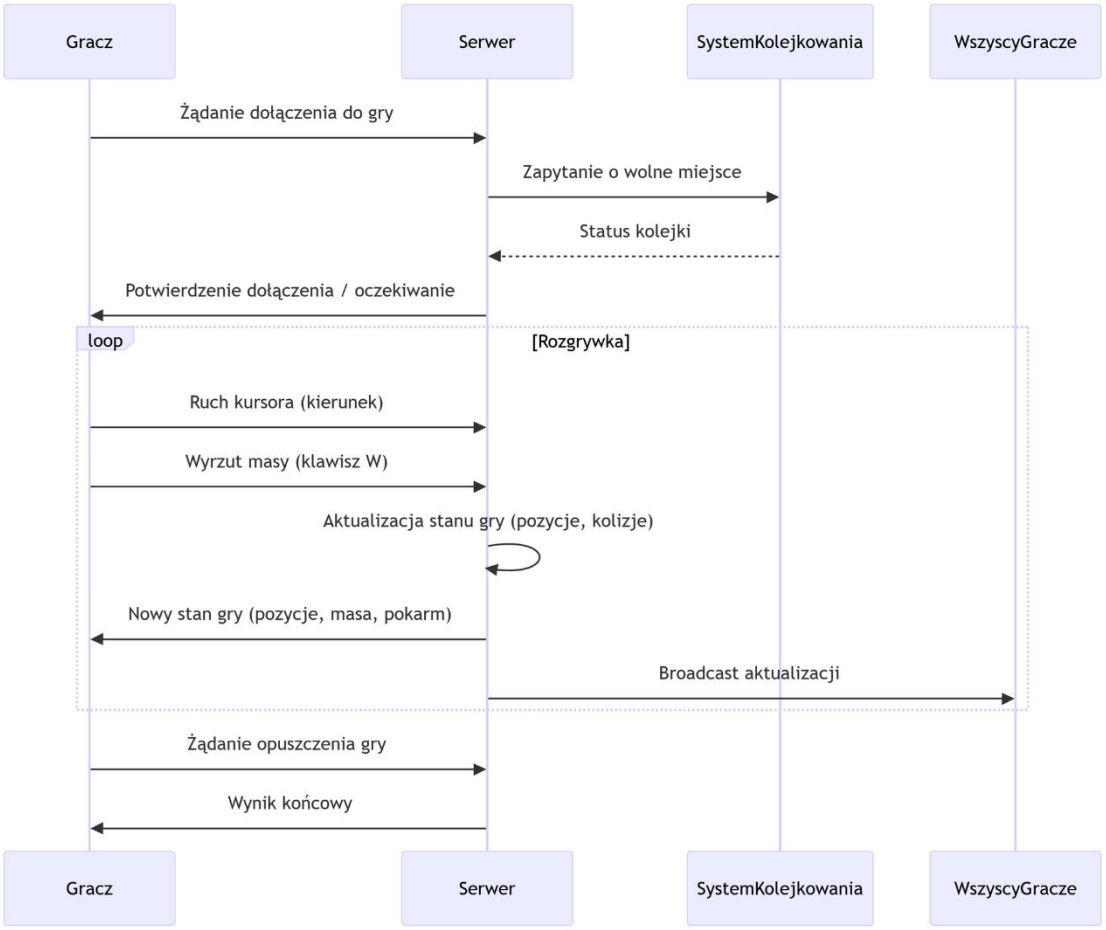
Część 2 – Diagramy sekwencji i klas:

Diagram sekwencji

Na tym diagramie sekwencji przedstawiony został przebieg interakcji pomiędzy graczem a serwerem w trakcie rozgrywki. Diagram odzwierciedla kolejność i logikę komunikacji w czasie rzeczywistym – od momentu dołączenia gracza do sesji gry, przez przesyłanie danych o ruchu i działaniach, aż po zakończenie połączenia. Jest to kluczowy element analizy systemu, ponieważ pokazuje, które komponenty biorą udział w wymianie informacji oraz jakie operacje są wykonywane.

W diagramie sekwencji można wyróżnić dwie istotne role: „Gracz" oraz „WszyscyGracze".

- „Gracz" reprezentuje pojedynczego użytkownika – jego interakcję z serwerem, przesyłane dane oraz odbierane odpowiedzi.
- „WszyscyGracze" natomiast to zbiorczy termin obejmujący wszystkich uczestników rozgrywki, do których serwer rozsyła zaktualizowany stan świata gry. Jest to uogólnienie, które podkreśla, że pewne informacje są rozsyłane zbiorczo do wszystkich aktywnych klientów, a nie tylko do pojedynczego gracza.

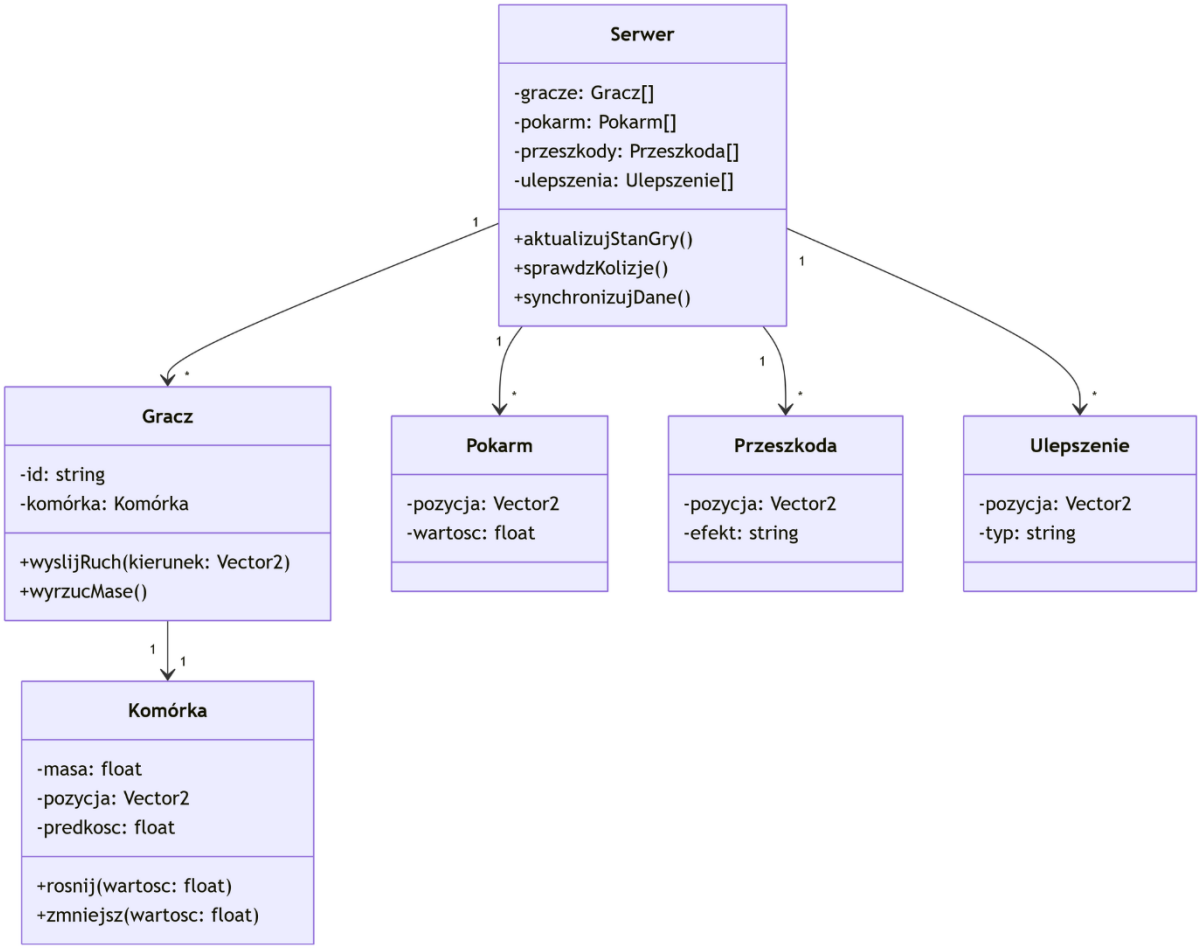


Opis diagramu:

- Gracz inicjuje połączenie z serwerem.
- Serwer sprawdza dostępność miejsca poprzez system kolejkowania.
- Podczas rozgrywki gracz wysyła dane o ruchu i wyrzucie masy, a serwer aktualizuje stan gry i rozsyła go do wszystkich uczestników.
- Kończy się rozłączeniem gracza i przekazaniem wyniku.

Diagram klas

Diagram klas przedstawia strukturę logiczną systemu gry CellFight w ujęciu obiektowym. Pokazuje on główne klasy tworzące system oraz zależności pomiędzy nimi. Model ten umożliwia zrozumienie, jak poszczególne komponenty są ze sobą powiązane i jakie pełnią funkcje w kontekście działania gry.



Opis diagramu:

Na powyższym diagramie klas wyróżniliśmy te klasy: Gracz, Serwer, Komórka, Pokarm, Przeszkoda i Ulepszenie. O to ich dokładniejszy opis:

GRACZ	Reprezentuje połączenie klienta z jego komórką. Odpowiada za identyfikację użytkownika oraz przekazywanie jego działań
SERWER	Zarządza wszystkimi elementami gry (gracze, pokarm, przeszkody, ulepszenia). Odpowiada za synchronizację stanu gry, kolizje, rozstrzyganie sporów oraz komunikację z klientami.
KOMÓRKA	Obiekt powiązany z graczem, przechowuje stan gracza (masa, pozycja, prędkość).
POKARM/PRZESZKODA /ULEPSZENIE	Obiekty na mapie z określonymi efektami.

Źródła

[1] https://agar.io/

[2] https://pl.wikipedia.org/wiki/Agar.io