Porównanie Wydajności Różnych Implementacji Symulacji Grawitacyjnej: Analiza i Optymalizacja

Szatan Arkadiusz 57562

Spis treści

[1. Wstęp 3](#_Toc167787998)

[1.1 Cel pracy 3](#_Toc167787999)

[1.2 Teoria 3](#_Toc167788000)

[2. Opis aplikacji 3](#_Toc167788001)

[2.1 Architektura aplikacji 3](#_Toc167788002)

[2.2 Opis funkcji i algorytmów 3](#_Toc167788003)

[2.3 Narzędzia i technologie 3](#_Toc167788004)

[3. Metodologia badania 3](#_Toc167788005)

[3.1 Kryterium wydajności 3](#_Toc167788006)

[3.2 Środowisko testowe 3](#_Toc167788007)

[3.3 Scenariusze testowe 3](#_Toc167788008)

[4. Wyniki i analiza 3](#_Toc167788009)

[4.1 Wyniki testów wydajnościowych 3](#_Toc167788010)

[4.2 Analiza Wyników 3](#_Toc167788011)

[5. Wnioski 3](#_Toc167788012)

# Wstęp

Symulacje grawitacyjne są nieodłącznym narzędziem w naukach przyrodniczych i technicznych, umożliwiając modelowanie i badanie dynamiki układów cząsteczek pod wpływem sił grawitacyjnych. Niniejszy projekt prezentuje implementację takiej symulacji w języku Python z wykorzystaniem biblioteki Pygame do wizualizacji.

## Cel pracy

Celem tego projektu jest porównanie wydajności dwóch wersji symulacji, z których jedna została zoptymalizowana pod kątem efektywności obliczeniowej.

## Teoria

Projekt składa się z kilku głównych komponentów:

* Symulacja grawitacyjna – modelowanie ruchu cząsteczek zgodnie z prawami fizyki.
* Detekcja kolizji – identyfikacja i obsługa zderzeń pomiędzy cząsteczkami.
* Interakcje użytkownika – możliwość dodawania nowych cząsteczek i manipulacji istniejącymi.
* Wizualizacja – przedstawienie symulacji w czasie rzeczywistym na ekranie komputera.

# Opis aplikacji

Aplikacja symulacji grawitacyjnej jest zbudowana z kilku kluczowych komponentów, które współpracują ze sobą, aby zapewnić realistyczną i wydajną symulację fizyczną.

## Architektura aplikacji

Architektura aplikacji obejmuje następujące elementy:

* Klasa Fluid - Reprezentuje pojedynczą cząsteczkę w symulacji. Każda cząsteczka ma swoje własne właściwości, takie jak pozycja, prędkość, promień, kolor oraz inne atrybuty potrzebne do symulacji.
* Klasa Vector - Reprezentuje wektor matematyczny używany do przechowywania pozycji i prędkości cząsteczek. Zawiera metody do operacji wektorowych, takich jak dodawanie, odejmowanie, skalowanie, normalizacja i obliczanie długości wektora.
* Moduł settings - Przechowuje wartości stałych używanych w aplikacji, takie jak kolory (WHITE), rozmiary okna symulacji (WIDTH, HEIGHT, SIZE), i inne ustawienia.
* Główna aplikacja (Game class) – Class jest główną klasą odpowiedzialną za zarządzanie symulacją, w tym inicjalizację środowiska, obsługę zdarzeń, aktualizację stanu symulacji, rysowanie elementów na ekranie oraz mierzenie wydajności.
* Detekcja kolizji (is\_collide function) - Funkcja sprawdza, czy dwie cząsteczki się zderzają. Jeśli tak, aktualizuje ich prędkości oraz pozycje zgodnie z zasadami fizyki.

## Narzędzia i technologie

Narzędzia i technologie wykorzystane w projekcie to głównie:

* Python: Język programowania, w którym napisany został cały projekt.
* Pygame: Biblioteka Pythona służąca do tworzenia gier i aplikacji multimedialnych. Wykorzystana do obsługi grafiki, dźwięku oraz interakcji z użytkownikiem.
* Biblioteki Pythona: math, time
* IDE PyCharm

# Porównanie wersji zoptymalizowanej i niezoptymalizowanej.

## Funkcja is\_collide

A screenshot of a computer program

Description automatically generated

Wersja niezoptymalizowana

A screen shot of a computer program

Description automatically generated

Wersja zoptymalizowana

## Funkcja sprawdza, czy dwie cząsteczki się zderzają. Jeśli tak, aktualizuje ich prędkości oraz pozycje zgodnie z zasadami fizyki. W zoptymalizowanej wersji, najpierw sprawdzamy czy odległość po x od kulek jest w wymaganym przedziale, jeżeli jest wtedy sprawdzamy czy odległość po y jest w wymaganym przedziale. Powinno to zoptymalizować obliczenia.

## Metoda update\_screen

## A computer code on a black background Description automatically generated

Wersja niezoptymalizowana

A screen shot of a computer program

Description automatically generated

A computer screen shot of a program code

Description automatically generated

Wersja zoptymalizowana

W wersji niezoptymalizowanej sprawdzana jest kolizja każdy do każdego, jest to sposób bardzo nie efektywny. Wraz ze wzrostem liczby kulek ilość sprawdzanych kolizji rośnie wykładniczo. W zoptymalizowanej wersji symulacji, przestrzeń ekranu jest podzielona na sektory (kwadranty), co umożliwia bardziej efektywne sprawdzanie kolizji. Cząsteczki są przydzielane do sektorów na podstawie swojej pozycji, co znacznie zmniejsza liczbę porównań potrzebnych do wykrycia kolizji.

# Metodologia badania

Metodologia porównawcza opierać się będzie na analizie ilości klatek na sekundę (FPS) dla różnej liczby kul w obu wersjach - zoptymalizowanej i niezoptymalizowanej. Proponowana metodologia składa się z następujących kroków

# Wyniki i analiza

## Wyniki testów wydajnościowych

|  |  |
| --- | --- |
| Wersja niezoptymalizowana | |
|  |  |
| Ilość obiektów | FPS |
| 0 | 60 |
| 10 | 60 |
| 20 | 60 |
| 30 | 60 |
| 40 | 60 |
| 50 | 60 |
| 60 | 60 |
| 70 | 60 |
| 80 | 60 |
| 90 | 60 |
| 100 | 60 |
| 110 | 60 |
| 120 | 57 |
| 130 | 50 |
| 140 | 42 |
| 150 | 37 |
| 160 | 33 |
| 170 | 30 |
| 180 | 27 |
| 190 | 23 |
| 200 | 22 |
| 210 | 20 |
| 220 | 18 |
| 230 | 16 |
| 240 | 15 |
| 250 | 14 |
| 260 | 13 |
| 270 | 12 |
| 280 | 11 |
| 290 | 10 |
| 300 | 9 |
|  |  |

|  |  |
| --- | --- |
| Wersja zoptymalizowana | |
|  |  |
| Ilość obiektów | FPS |
| 0 | 60 |
| 10 | 60 |
| 20 | 60 |
| 30 | 60 |
| 40 | 60 |
| 50 | 60 |
| 60 | 60 |
| 70 | 60 |
| 80 | 60 |
| 90 | 60 |
| 100 | 60 |
| 110 | 60 |
| 120 | 60 |
| 130 | 60 |
| 140 | 60 |
| 150 | 60 |
| 160 | 60 |
| 170 | 60 |
| 180 | 60 |
| 190 | 60 |
| 200 | 60 |
| 210 | 60 |
| 220 | 60 |
| 230 | 60 |
| 240 | 60 |
| 250 | 60 |
| 260 | 60 |
| 270 | 60 |
| 280 | 60 |
| 290 | 60 |
| 300 | 60 |
| 310 | 60 |
| 320 | 60 |
| 330 | 60 |
| 340 | 60 |
| 350 | 60 |
| 360 | 60 |
| 370 | 60 |
| 380 | 60 |
| 390 | 60 |
| 400 | 60 |
| 410 | 60 |
| 420 | 60 |
| 430 | 60 |
| 440 | 60 |
| 450 | 60 |
| 460 | 57 |
| 470 | 55 |
| 480 | 53 |
| 490 | 51 |
| 500 | 50 |
| 510 | 48 |
| 520 | 47 |

A graph with a line going up

Description automatically generated

## Analiza Wyników

Dla wersji niezoptymalizowanej punkt krytyczny, po którym następuje spadek wartości FPS jest przy 110 kulkach, a dla wersji zoptymalizowanej udało się przesunąć punkt krytyczny aż do 450 kulek. Tępo spadku FPS po przekroczeniu punktu krytycznego również udało się zredukować z 4.7 FPS na 10 klatek do 1.86 FPS na 10 klatek.

# Wnioski

Na podstawie przeprowadzonych testów i analizy wydajnościowej obu wersji aplikacji, można wyciągnąć następujące wnioski:

* Zoptymalizowana wersja aplikacji wykazuje znaczną poprawę wydajności w porównaniu do niezoptymalizowanej.
* Dzięki zastosowanym optymalizacjom, zoptymalizowana wersja osiąga stabilne i płynne działanie nawet przy dużej liczbie kulek na ekranie.
* Poprawienie punktu krytycznego po który FPS zaczynają spadać o 300%
* Redukcja tępa spadku klatek po przekroczeniu punktu krytycznego o blisko 60%.
* Pojawił się problem z zachowaniem kulek na łączeniu kwadrantów.