Uniwersytet Jagielloński
Wydział Matematyki i Informatyki
Zespół Katedr i Zakładów Informatyki Matematycznej

Wielowątkowa symulacja N ciał z implementacją w architekturze CUDA

Autor
Damian Stachura

Opiekun
dr Piotr Danilewski

8 czerwca 2018

Spis treści

1 **Wstęp** 3
- 1.1 Przedstawienie problemu symulacji N ciał . . . 3
- 1.2 Zastosowania . . . 4
- 1.3 Wykorzystanie CUDA w implementacji . . . 4

2 **Pierwsze podejście** 4
- 2.1 Jednowątkowy naiwny algorytm z pseudokodem . . . 4
- 2.2 Paralelizacja naiwnego algorytmu . . . 5
- 2.3 Implementacja . . . 5

3 **Drugie podejście** 5
- 3.1 Algorytm Barnesa-Huta z pseudokodem . . . 5
- 3.2 Zrównoleglenie algorytmu Barnesa-Huta . . . 5
- 3.3 Implementacja . . . 5

4 **Podsumowanie** 6

1 Wstęp

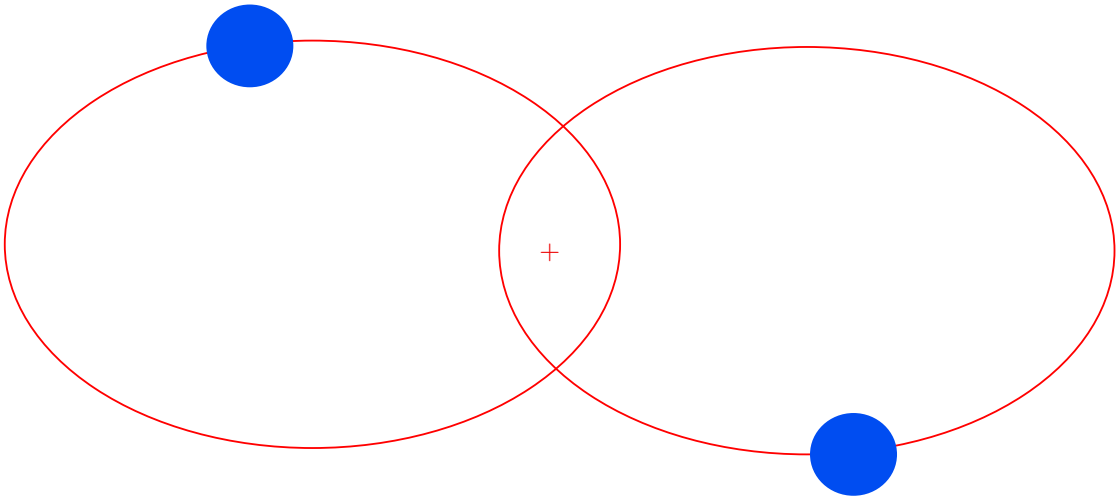
1.1 Przedstawienie problemu symulacji N ciał

Symulacja N ciał polega na wyznaczeniu toru ruchów wszystkich ciał danego układu o danych masach, prędkościach i położeniach początkowych w oparciu o prawa ruchu i założenie, że ciała oddziałują ze sobą zgodnie z prawem grawitacji Newtona.

Problem jest rostrzygnięty dla co najwyżej dwóch ciał podlegających prawom klasycznej dynamiki Newtona i przyciągających się zgodnie z newtonowskim prawem powszechnego ciążenia. Ruch ten wygląda w ten sposób, że obiekty poruszają się po krzywych stożkowych, przy czym rodzaj krzywej zależy od całkowitej energii układu. Jeżeli energia jest na tyle mała, że ciała tworzą stan związany, czyli nie mogą się od siebie uwolnić, tylko muszą krążyć wokół siebie, to robią to po elipsach. Jeżeli energia układu jest wystarczająco duża, żeby mogły oddalić się od siebie dowolnie daleko, to najpierw zbliżają się do siebie, a potem uciekają na ogół po hiperbolach, chyba że energia będzie dokładnie na granicy pomiędzy stanem związanym a niezwiązanym, kiedy to będą poruszać się po parabolach.
$http://www.deltami.edu.pl/temat/fizyka/mechanika/2015/11/26/Problem_dwoch_cial/$

Zajmijmy się stanem związanym. Ruch po elipsach jest okresowy, co oznacza, że co pewien ustalony okres ciała zajmują te same położenia w przestrzeni. Możemy wyobrazić sobie, że te dwa ciała to np. Ziemia i Słońce albo układ podwójny gwiazd. Jeżeli tylko ciała te mają symetrię sferyczną, to ich ruch jest dokładnie taki sam, jak ruch punktów materialnych o tych samych masach. Teoria Newtona mówi nam, że te dwa ciała będą krążyć po okresowych orbitach dowolnie długo.

Problem trzech ciał jest rozstrzygnięty jedynie dla szczególnych przypadków, w których zakłada się na przykład, że masa jednego z ciał jest zaniedbywalnie mała. Zagadnienie to – tak zwany ograniczony problem trzech ciał – było po raz pierwszy postawione i częściowo rozwiązane przez Lagrange'a w drugiej połowie XVIII wieku. Badał on układ Słońce-Ziemia-Księżyc, w którym na dodatek można przyjąć, że jedno z masywnych ciał krąży wokół trzeciego po orbicie kołowej.



Rysunek 1: Symulacja dwóch ciał po elipsie

1.2 Zastosowania

N-body simulations are widely used tools in astrophysics, from investigating the dynamics of few-body systems like the Earth-Moon-Sun system to understanding the evolution of the large-scale structure of the universe.[1] In physical cosmology, N-body simulations are used to study processes of non-linear structure formation such as galaxy filaments and galaxy halos from the influence of dark matter. Direct N-body simulations are used to study the dynamical evolution of star clusters. Common examples include a satellite orbiting a planet, a planet orbiting a star, two stars orbiting each other (a binary star), and a classical electron orbiting an atomic nucleus (although to solve the electron/nucleus 2-body system correctly a quantum mechanical approach must be used).

1.3 Wykorzystanie CUDA w implementacji

2 Pierwsze podejście

adadadadad

2.1 Jednowątkowy naiwny algorytm z pseudokodem

Listing 1: compute next step

```
void StepNaive::compute(tf3& positions, float dt) {
  std::fill(forces.begin(), forces.end(), 0);
  for(unsigned i=0; i<N; i++) {
    for(unsigned j=0; j<N; j++) {
      float distX = positions[j*3] - positions[i*3];
```

```
      float distY = positions[j*3+1] - positions[i*3+1];
      if(i!=j && fabs(distX) > 1e-10 && fabs(distY) > 1e-10) {
        float F = G*(weights[i]*weights[j]);
        forces[i*3] += F*distX/(distX*distX+distY*distY);
        forces[i*3+1] += F*distY/(distX*distX+distY*distY);
      }
    }
  }
  for(unsigned i=0; i<N; i++) {
    for(int j=0; j<2; j++) { // x, y
      float acceleration = forces[i*3+j]/weights[i];
      positions[i*3+j] += velocities[i*3+j]*dt + acceleration*dt*dt/2;
      velocities[i*3+j] += acceleration*dt;
    }
  }
}
```

2.2 Paralelizacja naiwnego algorytmu

asasasa

2.3 Implementacja

asasasa

3 Drugie podejście

aaaaaaaaaaaaaa

3.1 Algorytm Barnesa-Huta z pseudokodem

aaaaaaaaaaaaa

3.2 Zrównoleglenie algorytmu Barnesa-Huta

Huuuuut

3.3 Implementacja

Klepu klepu klep mammaamadmadadada Dummy text
apt-get install texlive-lang-polish
Random citation embeddeed in text.s
sudo apt-get install texlive-bibtex-extra
sudo apt-get install texlive-bibtex-extra biber
biber Praca
https://www.sharelatex.com/learn/Bibliography_management_with_biblatex
inkscape -D -z –file=drawing.svg –export-pdf=draw.pdf –export-latex

4 Podsumowanie

References

[Aar03] Sverre J. Aarseth. *Gravitional N-Body Simulations. Tools and Algorithms 1 edition.* Cambridge University Press, 2003.

[Cora] NVIDIA Corporation. *CUDA C Programming Guide.* v9.1.85, 2018. URL: http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html (visited on 03/05/2018).

[Corb] NVIDIA Corporation. *NVIDIA CUDA Runtime API.* v9.1.85, 2018. URL: http://docs.nvidia.com/cuda/cuda-runtime-api/index.html (visited on 03/05/2018).

[Gro17] Khronos Group. *OpenGL API, OpenGL Shading Lanugage and GLX Specifications.* OpenGL 4.6. 2017. URL: https://www.khronos.org/registry/OpenGL/index_gl.php (visited on 07/30/2017).

[Lar07] Jan Prins Lars Nyland Mark Harris. "GPU Gems 3". In: 2007. Chap. Fast N-Body Simulation with CUDA. Chapter 31, pp. 677–694.

[Lin99] Tancred Lindholm. "Seminar presentation. N-body algorithms". In: *University of Helsinki* (1999).

[Mar11] Keshav Pingali Martin Burtscher. "GPU Computing Gems Emerald Edition". In: NVIDIA Corporation, Wen-mei W. Hwu, 2011. Chap. An Efficient CUDA Implementation of the Tree-Based Barnes HUT N-Body Algorithm. Chapter 6, pp. 75–92.