



BERGISCHE
UNIVERSITÄT
WUPPERTAL

Galaxia - Ein Simulator für n -Körperprobleme

Maximilian Caspar und Johannes Esser

Bergische Universität Wuppertal

10. Juli 2016

Inhalt

■ Klassische Gravitationsphysik

- Das Gravitationsgesetz
- Das n -Körperproblem
- Algorithmen

■ Galaxia

- Sterne
- Galaxien
- Dateien

■ Simulationen

- Die Milchstraße
- Kollision von Milchstraße und Andromeda
- Homogene Verteilung



Das Gravitationsgesetz

- Das Newtonsche Gravitationsgesetz:
$$\vec{F}_g = G \frac{m_1 \cdot m_2}{r^2} \cdot \vec{e}_r$$
- In der ART hängt das Gravitationsfeld einer Masse unter anderem von ihrem Drehimpuls ab.
- Gravitation breitet sich mit c aus

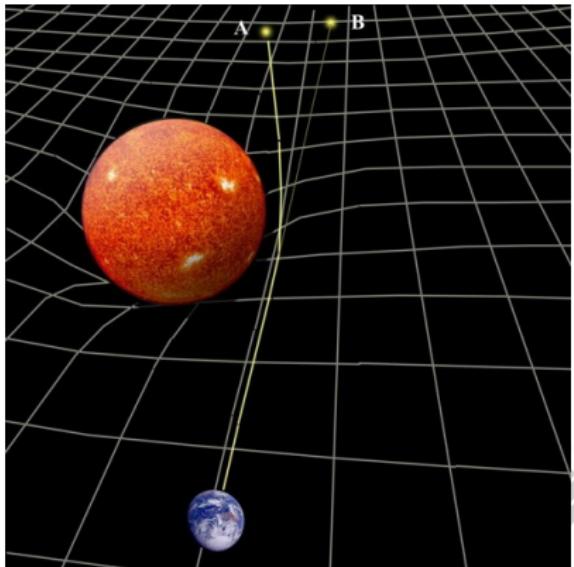


Abbildung: In der ART führt Gravitation zur Krümmung des Raumes (physicsoftheuniverse.com)

Das n -Körperproblem

- Das Verhalten von n Massestücken besitzt meistens keine analytische Lösung.
- Meist wird die Kollision von Galaxien oder die Entwicklung von Masseverteilungen betrachtet.



Abbildung: Ausschnitt aus dem
Millenium Run
(mpa-garching.mpg.de)

Algorithmus: Direkte Berechnung

Die klassische Berechnung eines Simulationsschrittes läuft so ab:

1. Berechne die Kräfte:
 - Wähle einen Stern.
 - Berechne die Kraft, die alle anderen Sterne auf diesen Stern auswirken und addiere diese auf.
 - Wähle einen anderen Stern. Fahre fort, bis die Kraft für alle Sterne berechnet wurde.
2. Bewege alle Sterne entsprechend ihrer aktuellen Geschwindigkeit.
3. Approximiere die neue Geschwindigkeit des Sterns nach
$$\vec{v}_{i+1} \approx \vec{v}_i + \frac{\vec{F}_i}{m_s} \cdot \Delta t$$
4. Setze die Kräfte auf die einzelnen Sterne auf $\vec{0}$.

Aufwand: $\mathcal{O}(n^2)$



Algorithmus: Barnes-Hut

1. Unterteile den Raum in einen Baum mit jeweils vier Knoten. Speichere Masse und Schwerpunkt jedes Knotens.
2. Durchlaufe den Baum rekursiv, für jeden Stern interagiere mit allen Knoten, die $\theta = \frac{d}{r} \leq \theta_0$ erfüllt.

Aufwand: $\mathcal{O}(n \log n)$

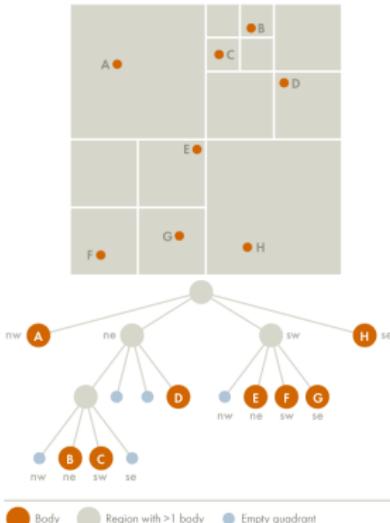


Abbildung: Ein Barnes-Hut-Baum (arborjs.org)

Galaxia

- Galaxia kann das Verhältnis zwischen beliebig vielen Sternen in einer Masseverteilung hinreichend gut simulieren.
- Das Programm ermöglicht die Darstellung einer oder mehrerer Galaxien und deren Eigenbewegung, sowie deren Kollision.
- Es eignet sich auch zur Untersuchung chaostheoretischer Phänomene.
- Masseverteilungen lassen sich über eingebaute Funktionen erzeugen, speichern und laden.



Sterne: Konstruktion der Klasse

Ein einzelner Stern speichert seine Masse, Position, Geschwindigkeit sowie die im aktuellen Schritt auf ihn wirkende Kraft.

```
Star(double sx, double sy, double vx, double vy,  
double m){  
    mass = m;  
    xpos = sx; ypos = sy;  
    xvel = vx; yvel = vy;  
    xfor = 0.0; yfor = 0.0;}
```



Sterne: Bewegung der Sterne

Der Stern wird auf Basis seiner momentanen Position, Geschwindigkeit und Kraft bewegt. Diese Funktion ist Mitglied der Klasse Stern.

```
void move(double dt){  
    xpos = xpos + xvel*dt;  
    ypos = ypos + yvel*dt;  
    xvel = xvel + (xfor / mass)*dt;  
    yvel = yvel + (yfor / mass)*dt;  
    xfor = 0.0;  
    yfor = 0.0;}
```



Galaxien: Erzeugung

Galaxien sind nichts anderes als Verteilungen von Sternen mit massereichem Zentrum.

```
while(rad < 0.05*radius){  
    x = distribution(generator); //Normalverteilung  
    y = distribution(generator);  
    rad = sqrt(x*x + y*y);}  
double alpha = get_angle(x , y);  
double v = orbit_velocity(rad , mco);  
Star temp(x + sx ,y + sy,v*sin(alpha) + vx,  
-v*cos(alpha) + vy,MASS_OF_SUN);  
Elements.push_back(temp);
```

Galaxien: Interaktion von Sternen

In Galaxia interagieren Sterne nach dem klassischen Berechnungsverfahren. Dieses ist genauer, aber auch langsamer. Wir wirken dem mit Parallelisierung des Codes entgegen.

```
#pragma omp parallel for
for(unsigned int i=0; i<n-1; i++){
    for(unsigned int j=i+1; j<n; j++){
        ...
        cluster[i].xfor -= con*xvec*pow(r, -3);
        cluster[i].yfor -= con*yvec*pow(r, -3);
        cluster[j].xfor += con*xvec*pow(r, -3);
        cluster[j].yfor += con*yvec*pow(r, -3);}}
#pragma omp parallel for
for(unsigned int i = 0; i < n ;i++){
    cluster[i].move(dt);}
```

Dateien: Laden und Speichern von Sternen

Vektoren, die Sterne enthalten, können geladen und gespeichert werden.

```
vector<Star> load(){      void save(vector<Star>&
    ifstream file{name};      out_stars){
    text_iarchive ia{file};    ofstream file{name};
    vector<Star> stars;        text_oarchive oa{file};
    ia >> stars;              oa << out_stars;}
    return stars;}
```

Dateien: Szenarien

Szenarien sind vorgefertigte Dateien mit allen nötigen Simulationsparametern.

```
gFile g("examples/scenarios/Collision1.galaxia");
elements = g.load();
for(unsigned int i = 1; i<=1000;i++){
    double ctime1 = omp_get_wtime();
    usleep(4);
    accelerate(elements, time_interval);
    export_plot(elements, 1.5E21);
    double ctime2 = omp_get_wtime();
    cout << ctime2 - ctime1 << endl;}
```

Simulation: Milchstraße



Simulation: Kollision von Milchstraße und Andromeda



Simulation: Homogene Masseverteilung

