Universiteit van Amsterdam

Inleiding Modelleren en Simuleren

Final Assignment

N-body simulator

David van Erkelens, 10264019

Department of Computer Science
University of Amsterdam

December 15, 2013

Alle code ook beschikbaar op
www.github.com/David1209/N-Body-Simulator

Contents

1 Inleiding **2**

2 Formules **2**

3 Implementatie **3**
3.1 Body-object 3
3.2 BodyPanel klasse 3
3.3 nBodySimulator 4

4 Experimenten **6**
4.1 Symmetrische planeten 6
4.2 Zonnestelsel 7

5 Beperkingen **8**

6 Conclusie **8**

1 Inleiding

Al sinds de tijd van de oude Grieken houdt men zich bezig met het bestuderen van de sterren. Met behulp van een N-body simulator kan de beweging van onder andere sterren, maar ook moleculen en andere lichamen welke invloed op elkaar uitoefenen gesimuleerd worden. In 1687 formuleerde Sir Isaac Newton de principes welke ten gronde liggen aan de aantrekkingskracht tussen twee lichamen. Hoewel Newton niet in staat was dit op de lossen voor meer dan twee lichamen, valt dit wel te benaderen met behulp van numerieke simulaties.

2 Formules

Om te berekenen welke invloed de verschillende lichamen op elkaar hebben, worden Newtons wetten van beweging en aantrekkingskracht gebruikt. Voor dit systeem wordt er vanuit gegaan dat voor ieder lichaam de beginpositie (p_x, p_y) en beginsnelheid (v_x, v_y) bekend zijn. Om vervolgens de dynamica van het systeem te kunnen bekijken, moet de kracht welke vanuit andere lichamen op het lichaam uitgeoefend wordt, worden doorgerekend. Om dit te kunnen doen, worden de volgende wetten van Newton toegepast:

- **Paarsgewijze kracht:** volgens Newtons wet van universele aantrekkingskracht wordt de aantrekkingskracht tussen twee lichamen beschreven door het product van hun massa's, gedeeld door het kwadraat van de afstand tussen hen:
$$F = \frac{Gm_1m_2}{r^2} \tag{1}$$
waarin G de gravitationele constante is, welke de waarde $G = 6.67 \times 10^{-11} N\frac{m^2}{kg^2}$ heeft.

- **Netto kracht:** volgens het principe van superpositie is de netto kracht op een lichaam gelijk aan de som van de paarsgewijze krachten van alle lichamen op het lichaam.

- **Versnelling:** volgens Newtons tweede wet van beweging wordt de versnelling van een lichaam gegeven door $a = \frac{F}{m}$.

3 Implementatie

Om de N-body simulator te implementeren, is er gebruik gemaakt van Java. Aangezien Java een object-geörienteerde programmeertaal is, is het heel logisch om voor ieder lichaam een apart object aan te maken.

3.1 Body-object

In `Body.java` wordt het object voor alle lichamen aangemaakt. Ieder lichaam heeft meerdere eigenschappen:

- Een (x, y) positie.
- Een (v_x, v_y) versnelling.
- Een massa.
- Een kleur om het lichaam in de simulatie te tekenen.
- De straal van het lichaam in de simulatie.
- Een `ArrayList` met de geschiedenis om een lijn achter het lichaam te kunnen tekenen.

Naast "getters" en "setters" voor bovenstaande eigenschappen, bevat de klasse nog de volgende functies:

- `update()`: deze voert een tijdstap door en verplaatst een lichaam naar een nieuwe locatie
- `getDistance(Body b)`: berekent de afstand tussen het huidige lichaam en meegegeven lichaam `b`
- Constructor `Body` waaraan de tekenstraal, locatie, versnelling en massa meegegeven worden.

3.2 BodyPanel klasse

Om de lichamen in de simulatie te kunnen tekenen, wordt de `BodyPanel` klasse gebruikt. Deze klasse is een uitbreiding van `JPanel`. Deze klasse bevat de volgende variabelen:

- `bodies`, een `ArrayList` met daarin alle lichamen welke getekend moeten worden (van de klasse `Body`).
- `worldsize`, de grootte van het universum waarin alle lichamen zich bevinden.
- `windowsize`, de grootte van het scherm waarin de simulatie uitgevoerd wordt.

Daarnaast bevat de klasse de volgende functies:

- `paintComponent`, een functie welke iedere keer als `.repaint()` aangeroepen wordt de nieuwe locatie van de lichamen tekent.
- `world2screen`, een functie welke het universum schaalt en transleert om in het scherm te passen.
- Constructor `BodyPanel` waaraan de `ArrayList` met lichamen, universumgrootte en schermgrootte meegegeven worden.

3.3 nBodySimulator

In `nBodySimulator.java` wordt de daadwerkelijke simulatie uitgevoerd. Omdat de `Body` klasse werkt met Cartesiaanse coördinaten, is het handig om de berekeningen van kracht en versnelling om te schrijven naar losse berekeningen langs de assen. Dit geeft:

$$F_x = F\frac{\delta x}{r} \tag{2}$$

$$F_y = F\frac{\delta y}{r} \tag{3}$$

voor de kracht, en

$$a_x = \frac{F_x}{m} \tag{4}$$

$$a_y = \frac{F_y}{m} \tag{5}$$

voor de versnelling. Omdat G een onafhankelijke constante is, wordt deze versimpeld naar de vorm:

$$G = 6.67 \times 10^{-11} \tag{6}$$

Nu deze gegevens bekend zijn, worden voor ieder lichaam F_x en F_y berekend. Hierbij moet opgemerkt worden dat het teken in δx en δy van belang is, deze afstanden worden berekend door de coördinaten van het huidige lichaam in mindering te brengen bij de coördinaten van het lichaam waarmee de kracht berekend wordt:

$$\begin{pmatrix} \delta x \\ \delta y \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix}_{anderlichaam} - \begin{pmatrix} x \\ y \end{pmatrix}_{huidiglichaam} \tag{7}$$

Nu F_x en F_y berekend zijn, kan de versnelling met behulp van formules 4 en 5 berekend worden. Er vanuit gaande dat de versnelling constant is in de grootte van de tijdstap, wordt de nieuwe versnelling van het lichaam gegeven door:

$$\begin{pmatrix} v_x + \delta t \times a_x \\ v_y + \delta t \times a_y \end{pmatrix} \tag{8}$$

Waarin δt is vastgesteld op 5×10^5. Aan de hand van deze versnelling kan de nieuwe locatie berekend worden. In `nBodySimulator.java` is dit als volgt geïmplementeerd (pseudocode):

```
for ieder lichaam:
    for iedere ander lichaam:
        berekend afstand;
        bereken force in x en y richting;
    tel alle x en y force bij elkaar op;
    bereken de nieuwe versnelling;
    stel de nieuwe versnelling in;

for ieder lichaam:
    verplaats het lichaam;
```

Het verplaatsen van het lichaam wordt in een nieuwe loop geplaatst, omdat eerst de oude positie van het lichaam voor ieder ander lichaam doorgerekend moet worden voordat het lichaam verplaatst kan worden.

4 Experimenten

Met de N-body simulator zijn twee experimenten uitgevoerd, een met twee symmetrische planeten welke om elkaar heen draaien en een met de helft van ons zonnestelsel.

4.1 Symmetrische planeten

Voor dit experiment zijn de volgende lichamen gebruikt:

	Startpositie	Startsnelheid	Massa	Kleur
Lichaam 1	$(0, 4.5e10)$	$(1.0e7, 0)$	$1.5e30$	Blauw
Lichaam 2	$(0, -4.5e10)$	$(-1.0e7, 0)$	$1.5e30$	Rood

Deze twee lichamen hebben dezelfde eigenschappen en blijven eeuwig om elkaar heen draaien, doordat deze telkens weer tot elkaar aangetrokken worden. Door de gelijke eigenschappen vertoont deze simulatie een symmetrisch uiterlijk. Dit experiment valt uit te voeren met het commando `java nBodySimulator two`.

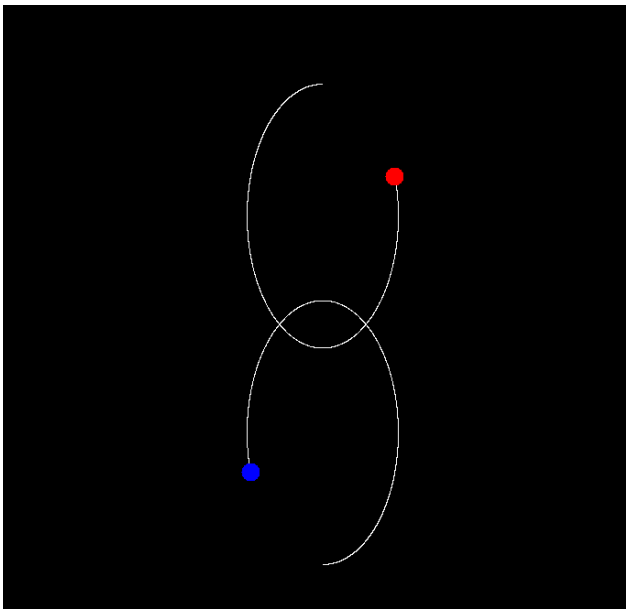

Figure 1: Twee planeten welke om elkaar heen draaien

4.2 Zonnestelsel

Voor dit experiment zijn de volgende lichamen gebruikt:[1]

	Startpositie	Startsnelheid	Massa	Kleur
Zon	$(0, 0)$	$(0, 0)$	$1.989e30$	Geel
Mercurius	$(5.79e10, 0)$	$(0, 4.79e7)$	$3.302e23$	Oranje
Venus	$(1.082e11, 0)$	$(0, 3.5e7)$	$4.869e24$	Roze
Aarde	$(1.496e11, 0)$	$(0, 2.98e7)$	$5.9740e24$	Blauw
Mars	$(2.279e11, 0)$	$(0, 2.41e7)$	$6,419e23$	Rood

In dit experiment komt de aantrekkingskracht van de zon mooi naar voren. Doordat de zon een enorme massa heeft in vergelijking met de andere planeten, blijft zelfs de meest ver weg liggende planeet (Mars) om de zon heen draaien. Ondanks dat de planeten onderling ook invloed op elkaar hebben, valt dit te verwaarlozen in vergelijking met de invloed van de zwaartekracht van de zon. Dit experiment valt uit te voeren door de simulator zonder argumenten te starten.

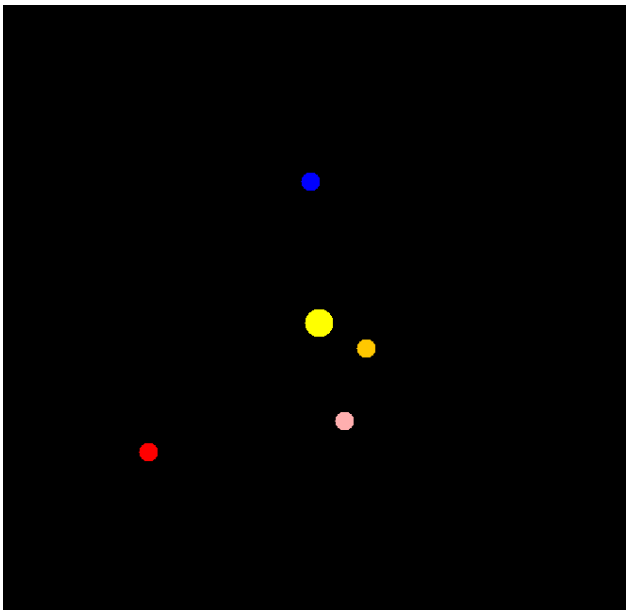

Figure 2: Het halve zonnestelsel

[1] Data afkomstig van www.cs.princeton.edu/courses/archive/fall13/cos126/assignments/nbody.html

5 Beperkingen

De grootste beperking van de N-body simulator is de preciezie. Doordat N-body simulaties voor $N > 2$ alleen te benaderen zijn en niet exact op te lossen, zal het resultaat van de simulator altijd een benadering zijn en nooit het exacte antwoord. Een verdere beperking in de implementatie van de N-body simulator valt te zien in het verschil tussen afbeelding 1 en 2. In afbeelding 1 worden er nog afgelegde banen achter de lichamen getekend, maar in afbeelding 2 niet. Dit valt te verklaren doordat de `ArrayList` waarin de geschiedenis van de lichamen bewaard worden na een tijd behoorlijk groot wordt, en hierdoor de simulatie zwaar vertraagd. Dit valt in de toekomst op te lossen door alleen het laatste stuk van de staart te tekenen of de staart met mindere precieze te tekenen.

6 Conclusie

Er valt te concluderen dat de N-body simulator goed werkt. Planeten laten duidelijk hun samenhang zien en de simulator loopt soepel. Ruimte voor verbetering is er nog door meer experimenten te doen en het probleem met het tekenen van de staart op te lossen. N-body simulaties blijken een redelijk natuurgetrouwe manier te zijn om samenhang tussen lichamen weer te geven, maar door het gebrek aan een volledige wiskunde oplossing zal het nooit een exacte beschrijving van de natuur zijn.