Galaxy Simulation

Jugend Forscht 2018/2019

Emile Hansmaennel
Theodor-Fliedner-Gymnasium, Heinrich Heine Universität Düsseldorf

2018 - 2019

Ist es möglich die Entstehung von Galaxien zu simulieren? Um diese Frage zu beantworten bin ich zu dem Schluss gekommen, dass ich das doch einfach mal ausprobieren sollte. Dazu habe ich das Navarro-Frenk-White Profil implementiert um anschließen die Kräfte die Zwischen den Sternen wirken zu berechnen. Dabei stattete ich die Sterne mit einer zufälligen Masse aus und Unterteilte die Galaxie in dynamisch-große Zellen um die Simulation stark zu beschleunigen. Um eine Stabile Galaxie zu simulieren müssen jedoch alle Sterne in der Galaxie eine Anfangsgeschwindigkeit besitzen die sie auf eine Kreisbahn lenkt, damit die Kraft, welche die Sterne in die Mitte der Galaxie zieht ausgeglichen werden.

Inhaltsverzeichnis

1 Einleitung

Das Projekt ist nach meinem vorletzten Jugend-Forscht Projekt entstanden: Ich habe ein Praktikum in Astronomischen Rechen Institut in Heidelberg genutzt, um mit einem Doktoranden[1] das Navarro-Frenk-White Profil, das zum generieren von Punkt Wolken genutzt wird, zu visualisieren. Anschließend hat sich das Projekt ein bisschen verlaufen, irgendwann beschloss ich jedoch, dass das Projekt weiterzuführen und statt nur statische Galaxien zu generieren dazu überzugehen die Galaxien zu simulieren, also die Entwicklung einer virtuellen Galaxie zu untersuchen. Eines der Entscheidenden Probleme war die Laufzeit der Simulation. Das Problem das es zu lösen galt, war die Nutzung von mehreren Threads mit der Nutzung des Barnes-Hut Algorithmus zu kombinieren. Das Ergebnis ist sehr schön: Durch die Nutzung der Programmiersprache Go war das einbauen der Nutzung von mehreren Threads vergleichsweise einfach.

2 Vorgehensweise

Wie schon in der Einleitung beschrieben habe ich mehrere Techniken kombiniert um mein Ziel zu erreichen. Das komplette Projekt lässt sich in mehrere Abschnitte unterteilen: Die Generierung der Punkt Wolke, aus der eine Galaxie abstrahiert wird. Die Simulation der Galaxie bei der die Kräfte die in der Galaxie wirken berechnet werden und daraus die Geschwindigkeit und Richtung in die der Stern fliegt.

3 Generieren

Das Generieren der Statischen Punkt Wolke aus der die Galaxie abstrahiert wird ist ein wichtiger Bestandteil des

[1] Tim Tugendhat

Gesamtprojektes, denn alles baut auf ihr auf. Kurz: um Kräfte zwischen Sternen zu berechnen braucht man erstmal Sterne!

3.1 Das Navarro-Frenk-White Profil

Das Navarro-Frenk-White Profil (NFW-Profil) ist ein Profil das genutzt wird, um die Räumliche Massen Verteilung von dunkler Materie anhand von Halos die in N-Körper Simulationen simuliert wurden, zu generieren. Das NFW-Profil kann jedoch auch genutzt werden um die Massen Verteilung von Sternen darzustellen. Das Profil generiert für einen gegebenen Abstand r zum Mittelpunkt der Galaxie eine Wahrscheinlichkeit ρ. Die Funktion sieht dabei wie folgt aus: NFW

$$\rho_{NFW}(r) = \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot\exp\left(\frac{-\phi(r)}{\sigma^2}\right) \tag{1}$$

$$\phi(r) = \frac{4\pi\cdot G\cdot f_0\cdot R_s^3}{r}\cdot ln\left(1+\frac{r}{R_s}\right)$$

Es kann nun mithilfe der Random-Sampling Methode (3.2) ermittelt werden, ob ein Stern beibehalten wird oder nicht.

Möchte man nun herausfinden wie weit ein Punkt mit der Koordinate $(x_1, x_2, x_3$ vom Mittelpunkt des Raumes entfernt ist, kann der Satz des Pythagoras (2) verwendet werden.

$$r_3 = \sqrt{x_1^2 + x_2^2 + x_3^2} \tag{2}$$

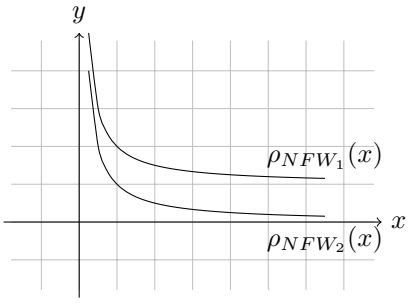
Der Abstand r_3 zum Mittelpunkt des Raumes kann nun in das NFW-Profil (1) gegeben werden um einen Wert s zu ermitteln welcher beim Random Sampling verwendet wird:

$$\rho_{NFW}(r) = \cdots = s \tag{3}$$

Dieser Wert s stellt die Wahrscheinlichkeit dar, das ein Stern der eine Entfernung r vom Mittelpunkt der Galaxie besitzt existiert.

Die Galaxie sieht aus der Ferne jetzt jedoch aus wie ein Würfel, da die aus ρ_{NFW_1} resultierende Kurve abrupt endet. Dies kann gelöst werden, indem statt $\rho_{NFW_1}(r)$ folgendes gerechnet wird: $\rho_{NFW_1}(r) - \rho_{NFW_1}(r_{max}) = \rho_{NFW_2}(r)$

Veranschaulichung:



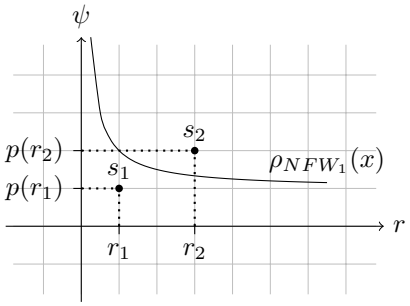
Problematisch ist hierbei die Tatsache, dass aufgrund der Verschiebung die Anzahl der Sterne die in Relation zu dem Bereich in dem sie generiert werden sehr stark sinkt.

3.2 Random Sampling

$$\psi = [\rho(r_{min}); \rho(r_{max})] \tag{4}$$

Sei s ein zufälliger Wert im Intervall ψ. Generiert man nun einen zufälligen Wert r im Intervall ψ, kann man schauen ob $s > r \vee s < r$ gilt. Ist $r > s$, wird der Stern verworfen, ist $r < s$ wird der Stern behalten.

Veranschaulichung:



In der obigen Abbildung ist zu sehen wir zwei zufällige Punkte s_1 und s_2 generiert wurden.

Angenommen es wurde ein Stern generiert für den nach Formel (2) gilt: $r = \sqrt{x_1^2 + x_2^2 + x_3^2} = \cdots = r_1$. Es wird dann ein Zufälliger Wert $p(r_2)$ im in (4) definierten Intervall generiert. Die folgenden zwei Fälle können dann eintreten und werden wie in (5) abgehandelt.

$$\begin{cases} s_1 \leq NFW(r_1) & \rightarrow \text{Stern wird beibehalten} \\ s_1 > NFW(r_1) & \rightarrow \text{Stern wird verworfen} \end{cases} \tag{5}$$

3.3 Lookup Tabellen

Statt nun für jeden Stern die Distanz des jeweiligen Sternes r in das NFW-Profil (1) einzusetzen, kann das NFW-Profil im vorhinein berechnet werden. Es wird dabei eine Tabelle erstellt in der die Entfernung des Sternes zum Mittelpunkt der Galaxie der jeweiligen Wahrscheinlichkeit zugeordnet wird:

$r_n \quad n \in \mathbb{N}$	$\rho_n \quad n \in \mathbb{N}$
r_1	ρ_1
r_2	ρ_2
r_3	ρ_3
$\cdots$	$\cdots$
r_n	ρ_n

Die Tabelle kann jedoch nicht so genaue Ergebnisse liefern wie das NFW-Profil, sie kann jedoch so angepasst werden, dass sie in den Arbeitsspeicher passt und somit das NFW-Profil so genau wie möglich widerspiegelt und das Generieren stark verbessert. Mit genügend Arbeitsspeicher ist der Fehler dann auch vernachlässigbar. Ein Kritischer Faktor, der beachtet werden muss wenn Lookuptabellen genutzt werden, ist die Geschwindigkeit des jeweiligen Speichermediums. Nutzt man z.B. Eine sehr langsame Festplatte kann es mehr Sinne machen die jeweiligen Werte direkt zu berechnen. Dagegen ist eine schnelle SSD (Solid-State-Drive) um einiges schneller.

TODO: Versuchsreihe.

3.4 Beschleunigung der Generierung

Es existieren mehrere Möglichkeiten die Generierung der Punkte zu verbessern.

Eine gute Möglichkeit ist die Nutzung von mehr Rechenleistung. Bei der Nutzung von n mal sovielen Rechen kernen ist das Generieren von Sternen n mal schneller. Die Server des Server-Hosters Hetzner können dabei gut verwendet werden: Es wird stündlich abgerechnet und 32 Kerne mit 128 GB RAM kosten $\approx$ 50ct/h was es ermöglicht für einen vergleichsweisen Günstigen Preis, sehr viele Koordinaten zu generieren.

Die Ausgabe von jeder Potentiellen Koordinate in die Kommandozeile verlangsamt die Generierung unglaublich stark, da der Rechner darauf wartet das die Ausgabe fertig ist bevor er mit der nächsten Rechnung beginnt was zu einer relativ starken Verlangsamung der Generierung führt.

4 Simulieren

4.1 Die Entstehung von Galaxien

"Eine Galaxie ist eine durch gravitation gebundene große Ansammlung von Sternen, Planetensystemen, Gasnebeln und sonstigen Stellaren Objekten." [2]

Demnach ist es relativ Einfach eine Galaxie zu generieren: es werden einfach ganz viele Objekte in einen Raum geworfen. Das reicht jedoch nicht um die Objekte als Galaxie definieren zu können, da sie nicht "durch Gravitation gebunden" sind.

Um dies zu tun muss die Kraft zwischen allen Objekten in der Galaxie berechnet werden um damit die Position der jeweiligen Objekte nach einer bestimmten Zeit bestimmen zu können.

Dies reicht jedoch auch nicht um eine "stabile" Galaxie zu generieren: berechnet man nur die Kräfte die auf ruhende Objekte in einem Reibungsfreiem Raum wirken, würden alle Objekte zum Massenmittelpunkt gezogen werden und die Galaxie würde somit implodieren. Es ist also nötig auf die Sterne in der Galaxie Anfangs Kräfte zu wirken. Diese Kräfte sind durch die Rotation der Galaxie um den Massenmittelpunkt der Galaxie definiert, man rotiert also die Galaxie und gleicht durch die Zentripetalkraft, die Kraft die Alle Sterne Richtung Massenmittelpunkt zieht aus. Rotiert man die Galaxie jedoch zu schnell, explodiert sie Förmlich, da die Sterne nicht mehr zusammengehalten werden und die Fliehkraft sie einfach auseinanderzieht.

4.2 Berechnung der Beschleunigungen

Um die Beschleunigung die auf einen Stern wirkt zu berechnen wird folgendes berechnet:

$$a = G \cdot \frac{\Delta M_2}{\Delta r^2} \tag{6}$$

G steht hier für die Universelle Gravitationskraft, ΔM für die Masse des Objektes das Umkreist wird und Δr für die Entfernung zum Mittelpunkt des Objektes das umkreist wird. Problem ist, dass kein Objekt umkreist wird sondern eine große Anzahl an Sternen. Es ist also nicht möglich mithilfe von herkömmlichen Methoden die Beschleunigung die auf einen Stern wirkt zu berechnen.

TODO: und jetzt?

4.2.1 Die Kraft als Vektor

Um die Kraft als Vektor darzustellen, muss die Formel 6 mithilfe von Vektoren wiefolgt neu aufgestellt werden:

$$\vec{F}_{12} = \underbrace{-G\frac{m_1 m_2}{|r_{12}|^2}}_{Scalar} \cdot \underbrace{\frac{r_2 - r_1}{|r_2 - r_1|}}_{Vector} \tag{7}$$

Die Summe der Kräfte die auf einen Stern wirken ist somit die Summe aller Kräfte die zwischen dem jeweiligen Stern a und allen anderen Sternen wirken:

$$F_a = \sum_{i=0}^{n-1} F_{ai} \tag{8}$$

4.2.2 Berechnung der Umlaufgeschwindigkeit

Die Umlaufgeschwindigkeit kann berechnet werden, indem die Kraft die die Sterne in die Mitte der Galaxie zieht $\left(F = G \cdot \frac{m \cdot M}{r^2}\right)$ mit der Zentripetalkraft $\left(F_z = \frac{m \cdot v^2}{r}\right)$ gleichgesetzt wird:

$$v = \sqrt{\frac{G \cdot M_E}{r}} \tag{9}$$

M_E steht dabei für die Masse der Erde, G für die Gravitationskonstante und r für den Bahn Radius. Da wir jedoch nicht in der Erdumlaufbahn, sondern in einer Galaxien Umlaufbahn hantieren, können wir nicht die Masse der Erde nutzen. Wir müssen daher eine andere Möglichkeit nutzen, die Größe der Masse, die den Stern in Richtung Massenmittelpunkt zieht zu berechnen.

4.2.3 Ellipsen und die Geschwindigkeit der Sterne

Da die Sterne nicht auf perfekten Kreisbahnen um den Mittelpunkt der Galaxie fliegen muss in betracht gezogen werden wie die Sterne auf Elliptischen Bahnen orbitieren. Wichtigt ist dabei die Geschwindigkeit, diese muss zwischen der ersten Kosmischen Geschwindigkeit v_k und der zweiten Kosmischen Geschwindigkeit v_P liegen. Die beiden Kosmischen Geschwindigkeiten sind folgendermaßen definiert:

$$v_{k1} = \sqrt{\frac{GM}{r}} \tag{10}$$

$$v_{k2} = \sqrt{\frac{2GM}{r}} \tag{11}$$

Die Tatsache das die Sterne auf Elliptischen Bahnen unterwegs sind ist für die Berechnung irrelevant, da eh für jeden Zeitschritt t eine neue Kraft berechnet wird aus der eine Beschleunigung berechnet wird die wiederum die neue Position des Sternes ergibt. Hält man die Geschwindigkeit der Sterne somit im interval $(v_{k1}; v_{k2})$, dann ergibt sich (in der Therorie) von alleine eine elliptische Bahn.

[2] `https://de.wikipedia.org/wiki/Galaxie`

4.3 Entwicklung der nötigen Software

Die Software ist komplett in Golang geschrieben was die nutzung von mehreren Threads mithilfe von Go-Methoden stark vereinfacht. Um den Barnes-Hut Algorithmus anzuwenden muss die Galaxie in einen Octree unterteilt werden. Dabei wird eine Zelle definiert die alle Sterne beinhaltet welche anschließen solange unterteilt, bis eine der drei Endbedingungen eintrifft.

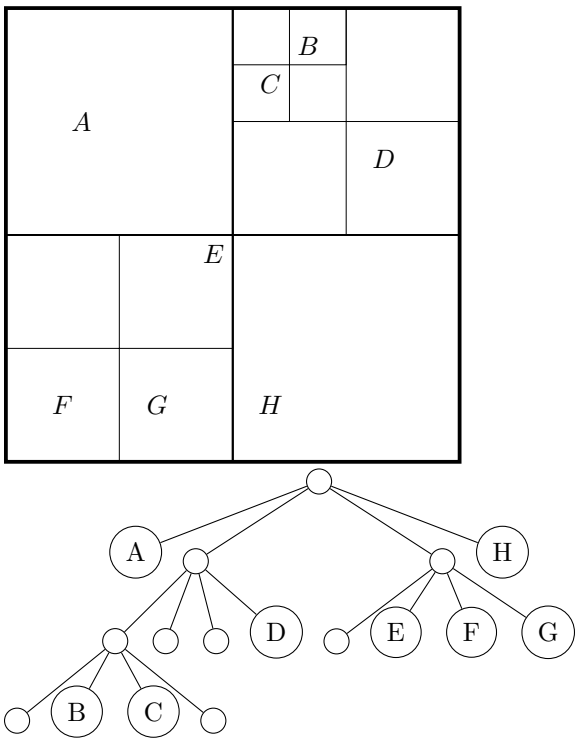
4.3.1 Zu lösende Probleme

Ein Problem das auftritt wenn die Kräfte zwischen allen Sternen berechnet werden ist, dass der Rechenaufwand $O(n \cdot n - 1) \approx O(n^2)$ beträgt.

Es kommt zu Problemen, wen nder mittlere Fehler, der bei der Berchnung der Kraft entsteht größer als die wirkende Kraft wird. Dies passiert unter anderem dann, wenn der Abstand zwischen den Sternen so groß wird, das die wirkende Kraft so gering ist das sie mithilfe von Computern nichtmehr sinvoll dargestellt wird. Statt nun mit Rundungsfehlen zu rechnen, können diese Sterne, die sehr weit enfrent vom Stern dessen kräfte berechnet werden sollen, einfach nicht mehr beachtet werden, da sie nicht sinvoll beitragen. Um dieses Problem zu lösen wird der Barnes-Hut Algorithmus verwenden. Dieser Algorithmus unterteil einen Bereich in variabel große Zellen und mindert die Laufzeit von $O(n^2)$ auf $O(n \log(n)$.

4.3.2 Generierung von Quadtrees und die nutzung des Barnes-Hut Algorithmus[3]

$$NW = \left(m \pm \frac{b}{2}, m \pm \frac{b}{2}\right) \tag{12}$$



[3]http://arborjs.org/docs/barnes-hut

Um die Kraft die auf einen bestimmten Stern wirkt zu berechnen, wird der Baum von der Wurzel aus nach unten durchlaufen. Beispiel: Berechnung der Kraft die auf den Stern A wirkt. Der 'Quadtree' wird von oben nach unten durchgegangen, also wird das Verhältnis zwischen der Entfernung des Massemittelpunktes zu dem Stern A (≈ 42) und der Breite der jeweiligen Zelle (100) berechnet (Formel 13): $\frac{100}{43} \not> \theta = 0.5$.

$$\theta = \frac{d}{r} \tag{13}$$

Ist das Verhältnis größer als ein im vorhinein definierter Grenzwert θ, dann wird weiter in den Quadtree ineingegangen und eine Ebene tiefer rekursiv weitergeprüft.

Die Koordinate des Massemittelpunktes ς des jeweiligen Knotens kann wie in Formel (14) beschrieben berechnet werden:

$$\varsigma = \left(\frac{\sum_{i=0}^{n} x_i \cdot m_i}{\sum_{i=0}^{n} m_i}, \frac{\sum_{i=0}^{n} y_i \cdot m_i}{\sum_{i=0}^{n} m_i} \right) \tag{14}$$

Es ist somit durch θ eine Endbedingung gegeben, welche verhindert das zu weit in den Baum vorgedrungen wird und somit auch verhindert, dass Sterne die in einer zu großen Entfernung zu dem ursprungssten liegen und dicht genug grupiert sind zusammengefasst werden.

4.3.3 Implementierung des Quadtrees

Um generell irgendwas zu tun muss in fast allen fällen etwas definiert werden. Zur Simulation von Galaxien brauchen wir vor allem eine Methode, Sterne einheitlich zu definieren. Der unten definierte Vec2-typ kann einen Vektor oder eine Koordinate darstellen was ihn in der Anwendung zu einem praktischen Hilfsmittel macht. Er speichert die X und Y Komponente der jeweiligen Struktur die er darstellen soll als float64-typ.

```
type Vec struct {
        X               float64
        Y               float64
        Z               float64
}

method (Vec2) insert() {...}
method (Vec2) insideOf() bool {...}
func newVec2() Vec2 {...}
```

Mithilfe des Vec2-typens kann ein Kompletter Stern definiert werden. Dabei wird ein Stern mithilfe seine Position C, seiner Geschwindigkeit V, und seiner Masse M beschrieben.

```
type Star struct {
        C               Vec
        V               Vec
        Mass            float64
}

method (Vec2) insideOf() bool {...}
```

Um einen sogennanten quadtree bzw. octree aufzubauen wird erstmal eine Räumliche Begrenzung benötigt, die einem Raum beschriebt indem die Sterne enthalten sind oder nicht. Diese grenze ist als 'Boundary' definiert, es wird dabei der Mittelpunkt der Begrenzung und die kürzeste Entfernung zwischen mittelpunkt und äußerer Begrenzung genutzt um den Raum zu definieren.

```
type Boundary struct {
        Center          Vec
        HalfDimension   float64
}

function newBoundary() {...}
```

Der eigentliche QuadTree bzw. Octree beinhaltet einige Informationen: Die Anzahl in ihm enthaltene Sterne, die Räumliche ausbreitung, die eigentlichen Sterne als Star2D definiert und die RecursionTiefe als integer. Die Definition des QuadTrees der Unten zu sehen ist enthält Zeiger zu den Kindern des Quadtrees und ist somit rekusriv definiert was es einfach macht neue Kinder zu erstellen, da diese eine Kopie ihere Eltern mit einer anderen Begrenzung darstellen wodurch die in ihnen enthaltenen Sterne weniger werden.

```
type QuadTree struct {
        StarCount       int
        Boundary        Boundary
        Star            []Vec2

        NorthWest       *QuadTree
        NorthEast       *QuadTree
        SouthWest       *QuadTree
        SouthEast       *QuadTree

        ReccursionDepth int
}
method (QuadTree) insert(Star) bool {...}
method (QuadTree) subdivide() {...}
```

4.3.4 Benchmarking

Um den Geschwindigkeitsvorteil darzustellen, kann die Kraft zwischen n homogen verteilten Sternen berechnet werden. Einmal mit der Brute-Force Methode und einmal mit der im Barnes-Hut Algorithmus beschriebenen Methode.

4.3.5 Runge-Kutta methods

Die Runge-Kutta Methode wird genutzt, um die Position eines objektes nach einer Beliebigen Zeit zu approximieren. Dabei kann, bei nutzung eines mglich kleinen Zeitschrittes, ein sehr genaues Ergebniss erzielt werden. In unserem Fall haben wir einen Stern auf den eine Kraft wirkt. Wir wollen die Position des Sternens nach einem Zeitschritt berechnen, jedoch auch eine andere Kraft miteinbringen um die Sterne auf eine Ellipische Bahn um die Mitte der Galaxie zu bringen. Die Geschwindigkeit die der Stern dabei annnimmt kann mit der fogenden Formel berechnet werden:

$$v = \sqrt{ar} \tag{15}$$

4.3.6 Goroutines

Die Nutzung von mehreren sogennanten Go-Methoden ist unglaublich effektiv, da es die Zeit die gebraucht wird das Programm auszuführen drastisch verringert. Die implementation ist ebenfalls unglaublich einfach, es recht

4.4 Netwerkfoo

Damit das Projekt so gut wie möglich skalliert, wird die Anwendung in mehrere kleine Dienste aufgeteilt die jeweils eine funktion übernehmen und untereinander kommunizieren. Dabei läuft jede Anwendung in einem eigenen Kontainer (siehe 4.4.2) und kann somit in falle eines nadelöhrs mehrfach gestartet werden und über einen reverse-http-proxy (siehe 4.4.4) mit daten versorgt werden.

4.4.1 Die verschiedenen Dienste

Um die Generierung der Punktwolken und die anschließende Simulation der Wolke in einzelne Microservices aufzuspalten muss erstmal definiert werden für welche Dienste es überhaupt sinn macht sie in einzelne instanzen abhängig bzw. unabhängig von einander agieren.

4.4.2 Docker

4.4.3 Docker-compose

4.4.4 Traefik

4.4.5 Kubernetes / Docker swarm

4.4.6 grafana