N-Body Simulation von Galaxien

Philip Späth und Kimi Sickinger

Dokumentation des Jugend forscht Projekts

„N-Body Simulation von Galaxien”

Betreut von Kai Marquardt und Manfred Brenner

Gewerbliche Schule Tübingen

Fachgebiet Mathematik / Informatik

Tübingen 2024

Projektüberblick

Im Rahmen unserer Forschungsarbeit haben wir eine Partikelsimulation entwickelt, mit der wir Galaxien, die sich bewegen und zusammenstoßen, betrachten und analysieren können. Es werden dabei Positionen und Massen der Sterne und die dadurch resultierenden Kräfte berechnet. Unsere verwendeten Algorithmen schließen die Betrachtung von Gravitation, Gaseigenschaften (Kräfte, Druck, Reibung) und die Expansion des Universums mit ein, was einen großen Rechenaufwand erforderlich macht. Um diesen zu reduzieren verwenden wir einen Ansatz, der als Barnes-Hut Algorithmus bekannt ist und als äußerst effizient gilt. Durch die Einbeziehung der genannten Faktoren erreichen wir eine sehr realistische Darstellung der Galaxien. Unsere Software simuliert verschiedene Galaxientypen, einschließlich spiralförmiger und elliptischer Galaxien, sowie ihre Verschmelzung.

Inhaltsverzeichnis

[1. Kurzfassung 1](#_Toc155962472)

[2. Idee und Motivation 1](#_Toc155962473)

[3. Grundlagen der Simulation 1](#_Toc155962474)

[4. Berechnung der Gravitationskraft 2](#_Toc155962475)

[5. Numerische Methoden 3](#_Toc155962476)

[5.1 Semi-implicit Euler Method (2. Order) 3](#_Toc155962477)

[5.2 Runge-Kutta Method (4. Order) 4](#_Toc155962478)

[5.3 Leapfrog Method (2. Order) 4](#_Toc155962479)

[5.4 Vergleich der Methoden 4](#_Toc155962480)

[6. Expansion des Universums / Hubble-Konstante 5](#_Toc155962481)

[7. Barnes Hut Algorithmus 5](#_Toc155962482)

[8. Software Struktur 6](#_Toc155962483)

[8.1 Synchronisierung der Berechnungen 7](#_Toc155962484)

[9. Simulation von Sonnen und Planetensystemen 8](#_Toc155962485)

[9. Simulation von elliptischen Galaxien 9](#_Toc155962486)

[10. Simulation von Spiralgalaxien 9](#_Toc155962487)

[10.1 Density Wave Theory 10](#_Toc155962488)

[11. Verteilung dunkler Materie 11](#_Toc155962489)

[12. Kollision der Milchstraße mit Andromeda (M31) 13](#_Toc155962490)

[13. Kollision der Whirlpool-Galaxie (M51) mit NGC 5195 14](#_Toc155962491)

[14. Fazit und Ausblick 14](#_Toc155962492)

[Literaturverzeichnis 14](#_Toc155962493)

# 1. Kurzfassung

Um Galaxien und ihre Kollisionen zu simulieren benötigt man eine n-body Simulation. Hierbei werden die Bewegungen der Körper über die Zeit berechnet. Anhand der Massen und Positionen werden im 3D Raum Kräfte wie die Gravitationskraft berechnet die dann auf die Körper wirken. Da die Beschleunigung der Körper nicht konstant ist, muss sie numerisch integriert werden. Dafür testeten wir verschiedene numerische Methoden und verglichen diese anhand des Energieverlusts im gesamten System (5.4 Vergleich der Methoden, Seite 4). Außerdem implementierten wir den Barnes-Hut Algorithmus der besonders bei vielen Körpern eine extrem große Reduzierung des Rechenaufwandes mit sich bringt. Bei diesem wird der Raum in viele kleine Würfel unterteilt, um die Masse bei weitentfernten Körpern zusammenzufassen. Da eine Spiralgalaxie im Schnitt ungefähr 100 Milliarden Sterne und unzählige Gaswolken enthält, ist es unmöglich alle diese unabhängigen Körper zu simulieren. Wegen der hohen Anzahl an Körpern arbeiten wir mit Dichteregionen, die in unserer Simulation als ein einziger Körper initialisiert werden. Beim Berechnen einer direkten Gravitationskraft bilden sich dann sehr schnell zu viele Cluster. Dies ist unrealistisch und kann verhindert werden, indem wir annehmen, dass sich diese Dichteregionen wie ein ideales Gas verhalten. Dafür verwenden wir die Smoothed Particle Hydrodynamics Technik, bei der zur Gravitationskraft noch eine Druck- und Reibungskraft addiert wird. Des Weiteren implementierten wir eine einfache Simulation der Dunklen Energie mit der Hubble-Konstante um die Expansion des Universums nicht zu vernachlässigen. Sie spielt besonders bei weit entfernten Galaxien durchaus eine Rolle. Mit unserer Simulation können nicht nur Elliptische Galaxien, sondern auch Spiralgalaxien simuliert werden. Bei diesen werden die Spiralarme gemäß der Density-Wave Theorie initialisiert. Das macht es möglich jegliche Galaxienkollision und -verschmelzung oder (mit genügend Rechenleistung oder Zeit) auch ganze Galaxienhaufen zu simulieren. Die Daten der Simulation können dann im Anschluss grafisch dargestellt werden. Um die Galaxien im 3D Raum auf einfache Weise rendern zu können, haben wir eine speziell für die Darstellung von Galaxien geeignete Engine geschrieben.

# 2. Idee und Motivation

Im Rahmen unserer Projekt Woche, am Ende des Schuljahres 22/23, in der wir ein Sonnensystem in 2d Simulierten wurde unser Interesse geweckt. Wir fragten uns ob wir eine solche Simulation auch in 3D umsetzten könnten und ob wir dann ganze Galaxien simulieren könnten. Uns interessierte besonders, was bei der Kollision von Galaxien passiert und wie diese zustande kommen. Da dies auch Grafisch sehr anschaulich gestaltet werden kann, sagte uns dieses Thema besonders zu da es nicht ganz so abstrakt ist, wie zum Beispiel die reine Simulation von Gasen oder Dunkle-Materie-halos war. Uns trieb dabei nicht nur das Interesse am Programmieren an, sondern auch die Neugier auf die Astrophysik hinter diesen aus unserer Sicht faszinierenden kosmischen Ereignissen.

# 3. Grundlagen der Simulation

Da die Daten unserer Simulation nicht in Echtzeit berechnet werden, wird der Prozess in zwei Hauptphasen gegliedert: die Berechnungs- und die Anzeigephase.

Während der Berechnungsphase werden Algorithmen genutzt, um die Bewegungen jedes einzelnen Körpers über die Zeit zu berechnen. Hierbei werden Gravitationskräfte nach Newtons Gravitationsgesetzt und andere Kräfte berücksichtigt, um so die Bewegung realistisch zu berechnen.

In der Anzeigephase werden diese berechneten Daten in einer 3D-Umgebung dargestellt. Nutzer können in dieser Phase durch die simulierte Galaxie navigieren, um die Bewegungen der Körper aus verschiedenen Blickwinkeln zu betrachten. Außerdem besteht die Möglichkeit, die Zeit in der Simulation vor- und zurück zu spulen, was einen detaillierten Einblick auf die Abläufe zwischen den Galaxien ermöglicht.

# 4. Berechnung der Gravitationskraft

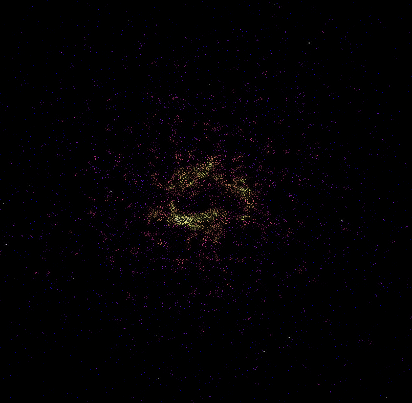
Um Graviton zwischen vielen unabhängigen Körpern zu berechnen muss die kraft zwischen jedem Objekt zu jedem anderen Objekt berechnet werden. Die sich daraus ergebende Dynamik wird als N-Body Simulation bezeichnet. Die Suche nach einer entsprechenden analytischen Formel wird in der Astrophysik als N-Body Problem [1] bezeichnet. Es ist ein bekanntes Problem welches nach heutigem Kenntnisstand nicht mathematisch lösbar ist. Außerdem ist die Beschleunigung der Körper nicht konstant weswegen numerische Methoden zur Annäherung nötig sind.

Um die Gravitationskraft, die auf einen Körper wirkt, auszurechnen benutzten wir Newtons Gravitationsgesetz:

Um eine Division durch 0, oder extrem große Kräfte bei zu kleinem Abstand zu verhindern glätten wir die Gravitationskraft unter Verwendung der Plummer Softening [2] Technik, wobei die Softening-Length ist und ein Skalierungsfaktor ist, der hier als Softening-Radius interpretiert werden kann:

Diese Implementierung der Gravitationskraft funktioniert hervorragend bei Festkörpern mit relativ großen Abständen wie zum Beispiel in unserem Sonnensystem. Da wir nun aber Galaxien simulieren wollen und nicht Milliarden von Sternen simulieren können, müssen wir mit Dichteregionen arbeiten und die Kraft zwischen diesen berechnen.

Erste Bilder zur dynamischen Clusterbildung durch unsere Software:



Nach 0 Jahren

Nach 500 Millionen Jahren

Nach 1 Milliarde Jahre

Durch die direkte Kraftberechnung bilden sich sehr starke Cluster. Um dies zu umgehen nehmen wir an, dass es sich bei unseren Dichteregionen um ein ideales Gas handelt. Um nun die Kräfte in diesem idealen Gas zu berechnen benutzten wir die Smooth-Particle-Hydrodinamics [3] Technik um die Gravitations-, Druck- und Reibungskraft zwischen den Dichteregionen realistisch abzubilden.

Um die Druckkraft zu berechnen müssen wir erstmal die Dichte für jeden Particle berechnen:

Mit der Dichte können wir dann den Druck berechnen, wobei die Referenzdichte ist:

Mit der Dichte und dem Druck können wir dann die Druckkraft berechnen, dabei ist der Gradient der Kernel-Funktion zwischen den Partikeln und :

Um nun auch noch die Reibung zwischen den dichten Regionen simulieren müssen wir zusätzlich noch die viskosen Kräfte berechnen, wobei der Laplacian der Kernel-Funktion zwischen den Partikeln und ist:

Die gesamte Kraft ist also:

Mit der Kraft die auf einen Körper wirkt, können wir seine Beschleunigung ausrechnen:

# 5. Numerische Methoden

Da die Beschleunigung a nicht konstant ist, müssen wir sie numerisch integrieren. Dazu gibt es etliche Methoden. Wir haben die weitverbeitesten getestet und untersucht. Dabei war uns vor allem die Rechengeschwindigkeit wichtig da wir nur begrenz Rechenleistung zur Verfügung haben.

## 5.1 Semi-implicit Euler Method (2. Order)

Wobei der Zeitschritt ist, bei dem wir annehmen die Beschleunigung sei konstant, da wir nur begrenzt Rechenleistung haben.

Die Euler Methode [4] ist die einfachste und auch die schnellste, doch sie ist auch recht ungenau, was schon nach einer kurzen Zeit eine extreme Auswirkung auf unsere Simulation haben kann.

# 5.2 Runge-Kutta Method (4. Order)

Die Runge-Kutta Methode(RK4) [5] ist deutlich genauer, doch sie ist auch deutlich rechenintensiver, was sie für unser Vorhaben ehr unpassend macht. Als wir am Anfang aber noch nur unser Sonnensystem simuliert haben, war sie am genausten und hat zum geringsten Energie Verlust geführt. Da die Runge-Kutta auch noch erweiterbar ist so dass sie Ordnung 8 oder mehr hat, könnten wir sie in Zukunft wieder einsetzen, wenn wir die Berechnungen an ein Rechenzentrum abgeben würden um somit genauere Ergebnisse zu erhalten.

## 5.3 Leapfrog Method (2. Order)

Kick Drift Kick (KDK):

Drift Kick Drift (DKD):

Die Leapfrog [6] KDK-Methode ist laut unseren Auswertungen am genausten, benötigt aber auch mehr Rechenleistung als die DKD-Methode, da zweimal die Beschleunigung ausgerechnet werden muss. Trotzdem haben wir für die Galaxien Simulationen meistens das KDK-Leapfrog Verfahren benutzt, da unsere Ergebnisse hiermit das beste Verhältnis zwischen Genauigkeit und Rechenintensivität hatten.

## 5.4 Vergleich der Methoden

Um die Genauigkeit der Methoden zu testen haben wir den Energie Verlust berechnet und diesen verglichen.

Die folgenden Energie Verluste wurden mit Daten von unserem Sonnensystem berechnet, da wir zur Zeit der Auswertung noch keine ganzen Galaxien simulieren konnten, sondern uns auf unser Sonnensystem beschränkt haben. Diese Daten haben wir vom NASA Horizons System [7], dass alle Positionen und Geschwindigkeiten von den Planeten in unserem Sonnensystem zu einem bestimmten Zeitpunkt bereitstellt.

Energieverlust der unterschiedlichen Methoden wobei die Achsen jeweils logarithmisch abgebildet sind

Wie in der Auswertung zu sehen schneidet KDK Leapfrog am besten ab Runge Kutta ist hier sehr ungenau, weil so groß ist, denn Runge Kutta wird erst ab einem sehr kleinem die beste Option. Nach dieser Auswertung haben wir für die finalen Berechnungen immer KDK Leapfrog als numerische Methode benutzt und zum schnellen Testen die Euler Methode.

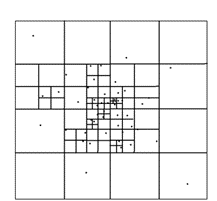
# 6. Expansion des Universums / Hubble-Konstante

Um auch die Expansion des Universums zu simulieren haben wir eine konstante Expansionsrate mit der Hubble-Konstante[[1]](#endnote-2) implementiert. Obwohl die Expansion des Universums ein hochkomplexes Thema ist und die Hubble-Konstante ja eigentlich nicht konstant, reicht diese Vereinfachung für die Zwecke unserer Galaxiensimulation aus.

Für unsere Simulation gehen wir von einer Hubble-Konstante [8] von  aus, wobei sie sich aus einem Mittelwert aus den zum Teil sehr unterschiedlichen Berechnungen der letzten 50 Jahre zusammensetzt.

# 7. Barnes Hut Algorithmus

Da die Gravitationskraft für jeden Körper in der Simulation mit jedem Körper berechnet werden muss, ist die Laufzeitkomplexität was zur Folge hat, dass größere und damit auch realistischere Mengen von Körpern zu einem quadratischen Anstieg des rechen Aufwands führen. Bei hunderttausend Objekten und zehntausend Zeitschritten wären es also schon 100 Billionen Berechnungen was mit heutigen Rechnern zu lange dauern würde. Um dieses Problem zu beheben, haben wir den Barnes Hut Algorithmus [9] implementiert. Mit diesem kann die Laufzeitkomplexität auf gesenkt werden, was einen riesigen Unterschied macht:

Die Umsetzung erfolgt in dem der Raum in Würfel unterteilt und alle Objekte in einer Baum Struktur eingeordnet werden. Dabei wird der Würfel, im Baum, hier als Node bezeichnet, so lange aufgeteilt bis sich auf der untersten Ebene in jeder Node nur ein Objekt befindet. Dieser Baum wird bei jedem Zeitschritt mit den aktuellen Objektpositionen neu generiert, damit die Würfel auch bei Verschiebungen die jeweils richtigen Objekte beinhalten.

Zusätzlich wird der Massenschwerpunkt der jeweiligen Node berechnet was die Genauigkeit der Berechnung um ein Vielfaches verbessert.

Unterteilung des Raums in Würfel

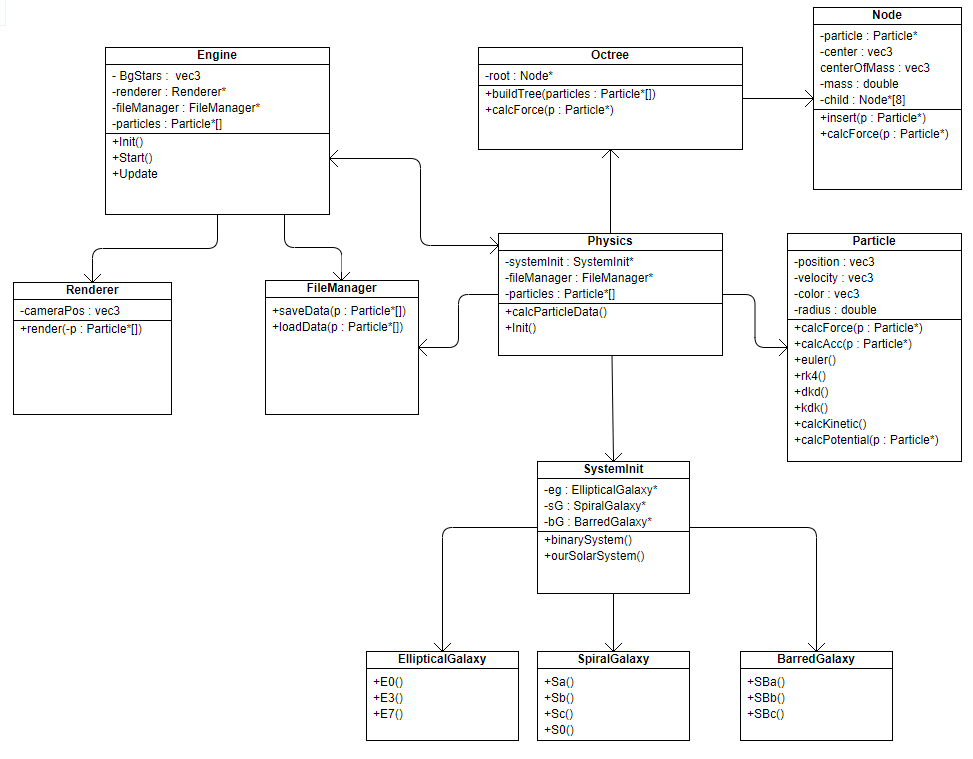
# 8. Software Struktur

Um die simulierten Daten auch entsprechend anzuzeigen haben wir uns für die Open Graphics Library entschieden. Dazu haben wir eine kleine Rendering Engine codiert, die die Partikel in 3D Raum in für uns geeigneter Weise anzeigen kann.

Verwendete Bibliotheken im Einzelnen:

* GLEW (OpenGL Extension Wrangler Library) für die vereinfachte Nutzung von OpenGL Funktionen.
* GLFW (Graphics Library Framework) für Fenster und Kontext Erstellung
* GLM (OpenGL Mathematics) für die Double Vektoren und Matrizen

Vereinfachtes UML-Diagramm mit den wichtigsten Klassen und ihren Attributen:



Zum Programmstart initialisiert SystemInit die Startpositionen und Geschwindigkeiten aller Objekte. Dabei können entweder unser Sonnensystem oder ganze Galaxien nach Hubbles Klassifikation [11] ausgewählt werden.

Nach der Initialisierung der Startpositionen berechnet Physics mit dem Barnes Hut Algorithmus(Octree) [9] die neuen Positionen für alle Körper und übergibt diesen dann an den File Manager, dieser speichert die Daten in Binärdateien. Die Engine kann sich anschließend die Daten zum gewünschten Zeitpunkt laden was uns die Möglichkeit gibt die berechnete Zeit vor und zurück zu spulen um die Veränderung im System besser zu verstehen.

Unser Gesamte Code ist Open Source und auf GitHub einsehbar, siehe Quelle [10].

# 8.1 Synchronisierung der Berechnungen

Um den Prozess der Berechnungen zu beschleunigen, wurde Multithreading ins Programm eingebaut. Multithreading ist die synchrone Verteilung der Berechnungen und abzuarbeitenden Aufgaben auf alle Kerne des Prozessors.

Ein normaler PC hat im Durchschnitt 4-8 Kerne. Ein Server hingegen hat häufig 128 oder mehr Kerne, was die Berechnung um ein Vielfaches beschleunigt. Hochleistungsserver kommen sogar auf über 1000 Kerne.

Ein weiterer Ansatz wäre es die Berechnungen auf die Grafikkarte zu übertragen, da diese auf bestimmte Berechnungen spezialisiert sind.

Eine weitere Überlegung war die Berechnungen auf mehrere Computer (Gridcomputing) zu verteilen.

Beide dieser Ansätze wurden aber bis her noch nicht weiterverfolgt, da die Variante mit einem Server am schnellsten und einfachsten umzusetzen war.

Um zu testen, ob und gegebenenfalls wie viel schneller die Berechnungen mit Multithreading sind, wurden Tests angestellt. Es wurde die Benötigte Zeit für eine Berechnung mit und ohne Multithreading gestoppt. Diese Berechnungen haben wir mit verschiedener Partikelzahl und unterschiedlichen Zeitschritten getätigt. Da die Galaxien zufällige Startwerte haben, wurde für jede generierte Galaxie Berechnungen mit und ohne Multithreading angestellt. Um das Fehlerpotenzial zu minimieren, wurde für jede Einstellung der Durchschnittswert aus 5 Berechnungen genommen. Wir haben außerdem auf keine oder nur unvermeidbare Hintergrund Prozesse geachtet.

An diesem Diagramm ist zu sehen, dass bereits ab 1000 Zeitschritten und 10000 Partikeln sich die Rechenzeit mehr als halbiert. Die Rechenzeit verkürzt sich mit diesen Daten um den Faktor. Weitere Messungen haben ergeben, dass die mögliche Beschleunigung durch Multithreading von der Anzahl von Zeitschritten und Partikeln abhängig ist. Bei einer zu geringen Partikel Anzahl ist Multithreading langsamer, da das Aufteilen und die Verwaltung der Threads Zeit in Anspruch nimmt. Der Schwellwert liegt bei ca. 100 Partikeln.

Vergleich Multithreading und kein Multithreading bei unterschiedlich vielen Zeitschritten bei 1000 Partikeln.

Vergleich anzahl der Kerne bei Multithreading in bezug zur Zeit.
Um zu testen, welchen Vorteil eine Vergrößerung der Kerne Anzahl bringt, wurden die gleichen Tests auch noch mit zwei Kernen angestellt. Leider können wir keinen weiteren Test mit sechs oder acht Kernen machen, da wir dafür keine entsprechende Hardware haben.

Wie in dem Diagramm zu erkennen ist, ist die Zeit linear ansteigend zu der Anzahl der Kerne.

Vergleich Multithreading mit unterschiedlicher Anzahl an Kernen in Bezug auf Zeit bei 10000/20000 Partikeln und 1000 Zeitschritten.

# 9. Simulation von Sonnen und Planetensystemen

Angefangen haben wir mit einer einfachen Simulation unseres Sonnensystems. Der Aufbau dieser Simulation ist grundlegend der gleiche wie bei einer größeren und komplexeren Simulation ganzer Galaxien. Allerdings gibt es einige Vereinfachungen.  
Somit wird kein Barnes Hut Algorithmus gebraucht, da es im Vergleich zu Galaxien ein minimaler Rechenaufwand ist, welcher auch ohne Barnes Hut Verfahren schnell berechnet werden kann. Des Weiteren wird kein SPH benötigt, da der Abstand im System zu groß und die Partikelanzahl zu gering ist und somit das System nicht als Gas betrachtet wird.

# 10. Verifizierung unserer Software

## 10.1 Verifizierung gegen die Gesamtenergie

Unsere erste Simulation bestand nur aus 2 Objekten. Einer Sonne als Zentrum und einem in Masse und Geschwindigkeit angepassten Planeten. Nach der oben beschrieben Berechnung (5.4 Vergleich der Methoden) hatten wir bei dieser Simulation nach einer berechneten Zeit von 2 Jahren eine Energieverlust von 0,5%.

## 10.2 Verifizierung gegen NASA Daten

Um unsere eigenen Simulationsergebnisse zu überprüfen haben wir vom Horizons System der NASA [7] die Daten unseres Sonnensystems aktuell, vor einem Jahr und von vor zwei Jahren angefordert. Das Horizons System liefert die genauen Positionen und Geschwindigkeiten aller Objekte in unserem Sonnensystem.  
Somit konnten wir unsere Berechnungen des Sonnensystems mit den Daten der NASA vergleichen:

Startwert unserer Simulation waren die NASA-Daten von vor zwei Jahren. Nach den von uns berechneten Planetenpositionen nach einer Zeit von einem Jahr hatten wir eine Abweichung von 0,3% zu den Werten der von der NASA gemessenen Positionen. Nach zwei Jahren hatten wir 0,6% Abweichung.

Damit war eine gute Basis für die Simulation von kompletten Galaxien geschaffen. Anhand der Vergleiche von Energieverlust und den NASA Daten wissen wir jetzt, dass unsere Simulation physikalisch korrekt ist und in der Zukunft gut erweitert werden kann.

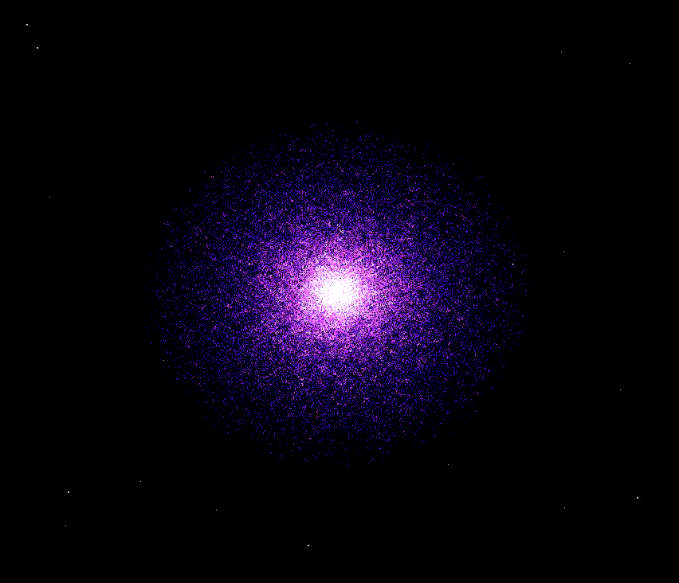
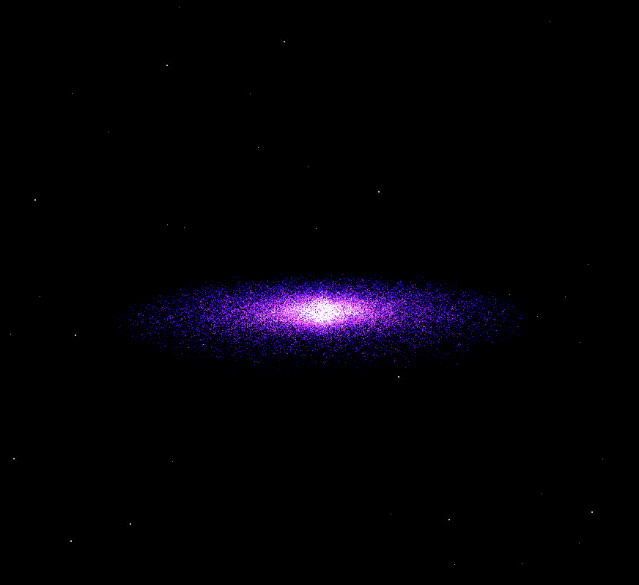
# 11. Simulation von elliptischen Galaxien

Anfangs haben wir uns auf Elliptische Galaxien fokussiert, da sie wegen ihrer fehlenden Spiralarme einfacher zu initialisieren sind.

Die Gesamtmasse, Größe und dichte Verteilung kann individuell angepasst werden. Es kann also jede beliebige elliptische Galaxie initialisiert werden um sie zum Beispiel mit einer spiralen Galaxie oder anderen Typen verschmelzen zu lassen.

Dabei haben wir 5 Standard elliptische Galaxien nach der Hubble Klassifikation [11] integriert (E0, E3, E7 und S0).

Unsere Ergebnisse:

 E7 Galaxie (Hubble Klassifikation [11]): S0 Galaxie (Hubble Klassifikation [11]):

Kugelförmige elliptische Galaxie mit der Größe, Gesamtmasse und Verteilung von Messier 89

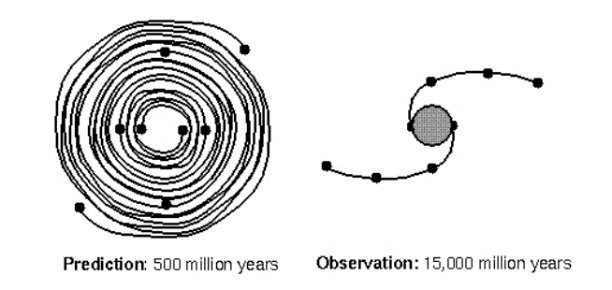
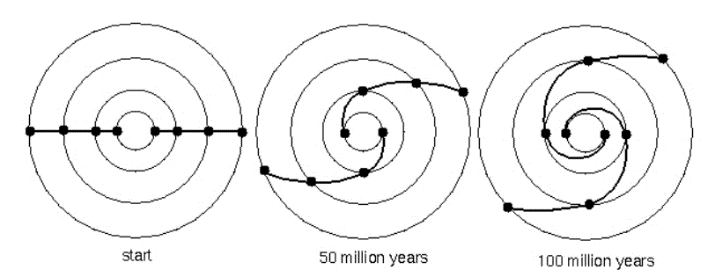
Scheibenförmige elliptische Galaxie mit der Größe, Gesamtmasse und Verteilung von NGC 3156

# 

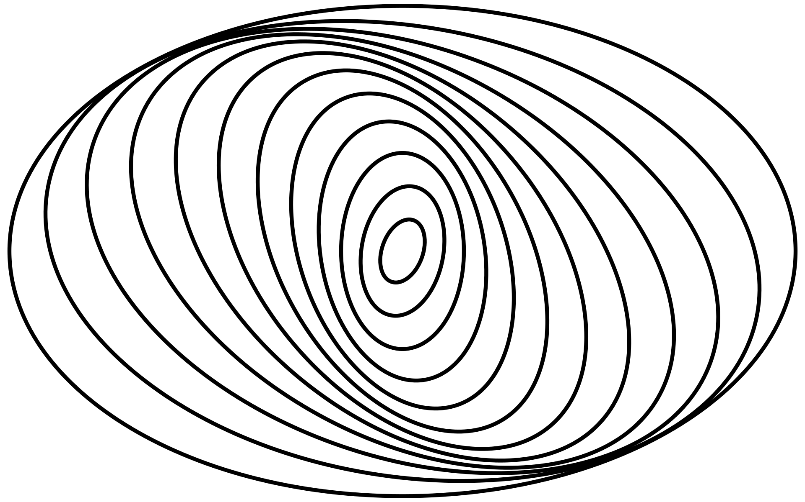
# 12. Simulation von Spiralgalaxien

Um nun Spiralgalaxien zu simulieren, müssen wir neue Ansätze verfolgen, da die Masse nicht einfach bei allen Winkeln gleich verteilt ist, sondern es Spiralarme gibt, in denen mehr Masse ist. Wenn man versucht die Startpositionen der Sterne so festzulegen wie sie beobachtbar sind und sie dann auf einem relativ runden Orbit um das Zentrum kreisen lässt wie bei der elliptischen Galaxie. Dann stößt man relativ schnell auf ein Problem: Auch wenn wir die Spiralarme am Anfang richtig initialisieren, werden sich diese schon nach wenigen Umläufen komplett auflösen.

Dieses Problem wird in der Astronomie Winding Problem genannt:



https://faculty.washington.edu/ivezic/Teaching/Astr509/lecture12.pdf

Um dieses Problem zu lösen haben C.C. Lin und Frank Shu Mitte der 1960er die Density Wave Theory entwickelt, um die Bewegung von Spiralarmen zu beschreiben.

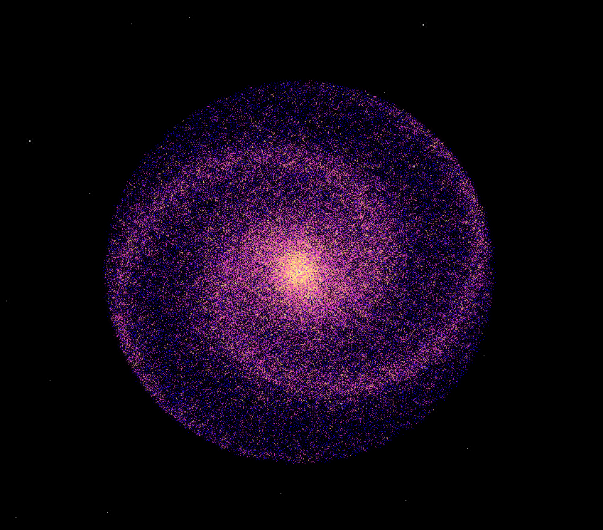
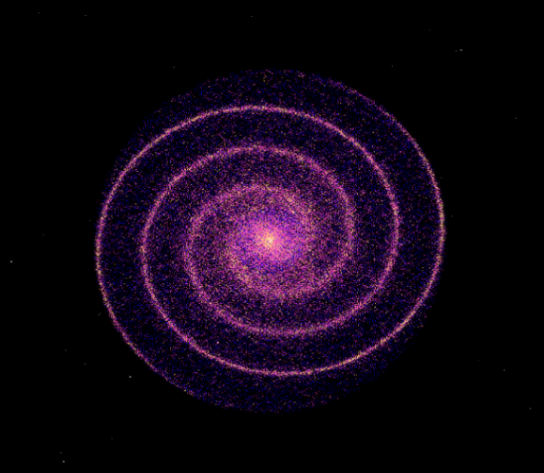
## 10.1 Density Wave Theory

Sie nimmt an, dass in den helleren Regionen der Galaxie die Spiralarme zu sehen sind, da sich an diesen Stellen die Orbits am nächsten sind und dadurch Dichtewellen entstehen. Um dies zu erreichen müssen alle Orbits von innen nach außen immer ein bisschen gedreht werden um diese Dichtewellen zu erzeugen.

https://en.wikipedia.org/wiki/Density\_wave\_theory

Von uns generierte Bilder:

Sc Galaxie (Hubble Klassifikation [11]): Sb Galaxie (Hubble Klassifikation [11]):



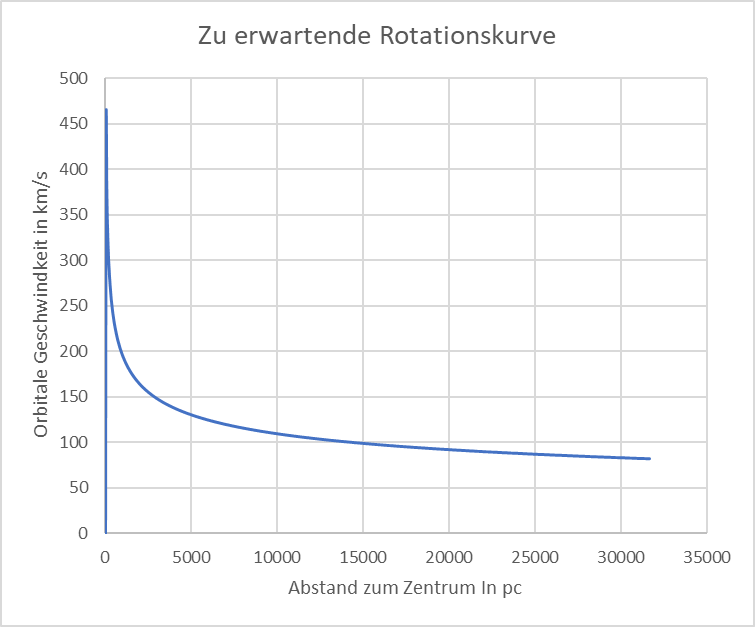
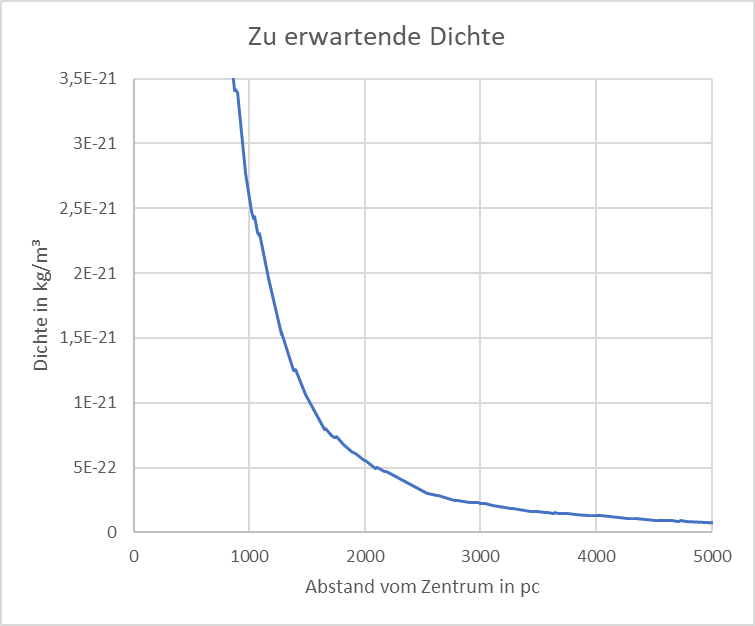
Extrem stark spiralförmige Galaxie mit der Größe, Gesamt Masse und Verteilung von Messier 51

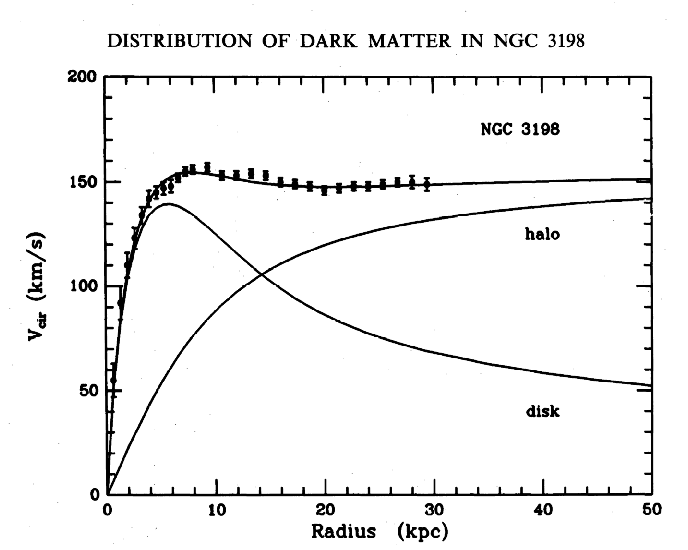
spiralförmige Galaxie mit der Größe, Gesamt Masse und Verteilung von NGC 1300

# 11. Verteilung dunkler Materie

Aufgrund verschiedener Beobachtungen ist es wissenschaftlicher Konsens, dass dunkle Materie existiert und einen Großteil der Materie im Universum ausmacht. Eine wichtige Rolle spielen dabei Rotationskurven von Galaxien. Um die Bedeutung dieser Kurven zu erkennen müssen wir erst verstehen wie die Orbitalen Geschwindigkeit überhaupt zu Stande kommen.

Damit der Stern auf einem stabilen Orbit, bleibt muss die Gravitationskraft der Zentripetalkraft entsprechen:

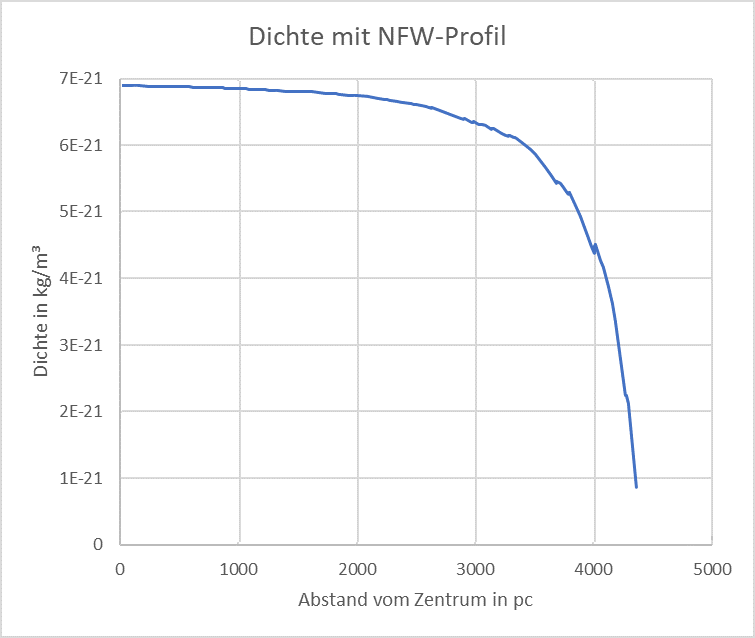
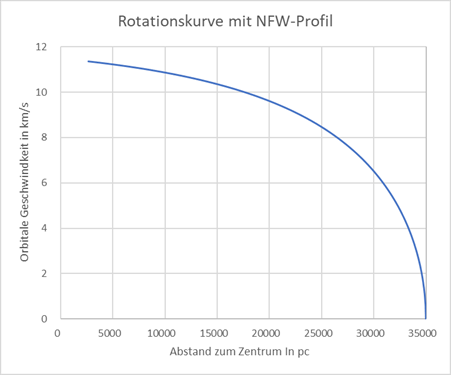
Wenn also die Dichte im Zentrum größer ist, wie es auch Astronomen lange vermutet haben, sieht die Rotationskurve so aus (berechnete Daten aus unserer Software):

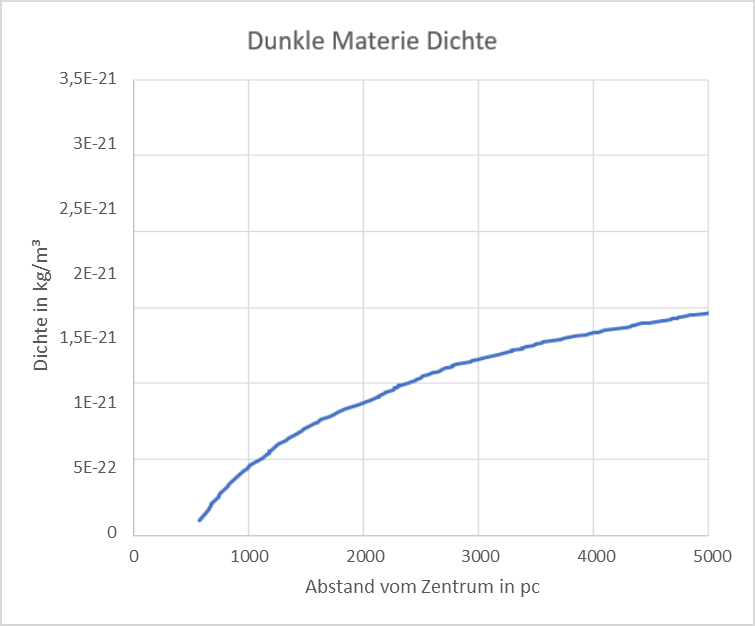
 Jedoch zeigen Beobachtungen, dass die Rotationskurven vieler Galaxien anders aussehen: Nach einem initialen Anstieg bleiben die Rotationsgeschwindigkeiten über weite Bereiche außerhalb des sichtbaren Teils der Galaxie nahezu konstant oder fallen nur sehr langsam ab. Um eine flache Rotationskurve zu erreichen muss die Masse innerhalb des Orbits proportional zum Abstand vom Galaxiezentrum sein:

Rotationskurve von der Galaxie NGC 3198

Diese zusätzliche nicht sichtbare Masse lässt den Schluss zu, dass es sich um eine andere Materie als unsere baryonische handelt, die nicht aus Elektronen, Neutronen und Protonen besteht. Diese nicht mess- oder sichtbare Materie wird Dunkle Materie genannt. Die notwendig ist um die Rotationskurven von Galaxien zu erklären.

Eine populäre dichte Verteilung ist das Navarro–Frenk–White profile (NFW) [12] welches sowohl aus N-Body Simulationen als auch aus Beobachtungen entstanden ist:

Wobei die Dichte ist, und eine Referenzdichte und der Galaxieradius sind. Unsere berechneten Werte:

Um nun eine wirklich realistische Massenverteilung und realistische Rotationskurve umzusetzen müssen wir ein Dunkle-Materie-Halo, dass im äußeren Bereich der Galaxie dichter ist, hinzufügen. Die gesamte Masse muss also nach außen ungefähr proportional zum Abstand zunehmen.

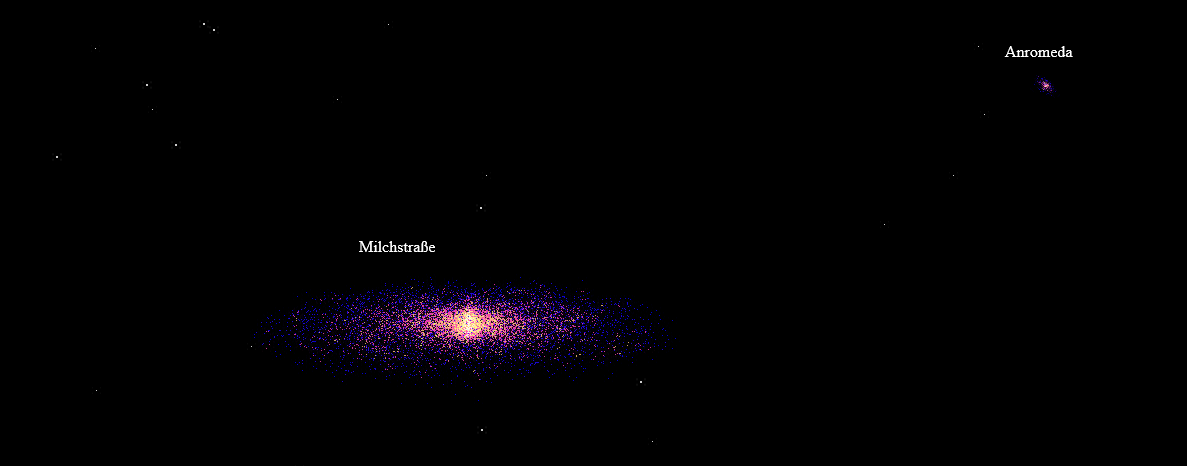
Dabei macht der Dunkle Materieanteil ungefähr 90% der Gesamt Masse aus.

Rotationskurve mit NFW-Profil [12] und von uns ergänzter Dunkler Materie:

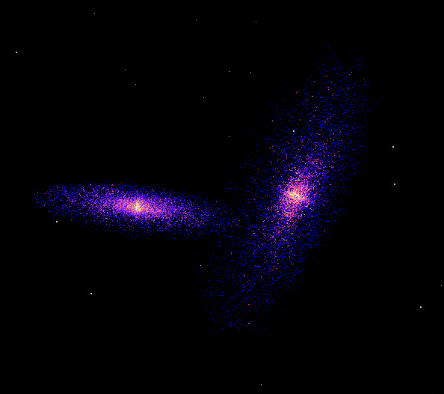
# 12. Kollision der Milchstraße mit Andromeda (M31)

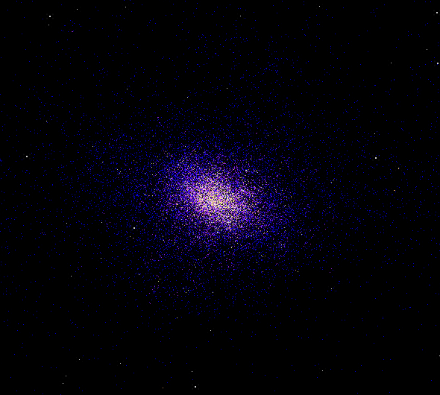
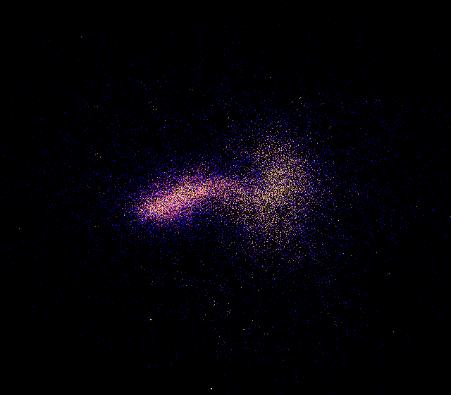
Als Anwendungsbeispiel haben wir die Kollision der Milchstraße und der Andromeda Galaxie simuliert. Wichtig ist jedoch zu betonen, dass jegliche Galaxien Konstellationen simuliert werden können, wenn Daten über Abstände, Durchmesser und Massenverteilung vorhanden sind.

Das Zentrum der Andromeda Galaxie [13] ist 2.51 ± 0.13 Millionen Lichtjahre vom Zentrum der Milchstraße entfernt. Ihre Masse wird auf 1,5 Billionen Sonnenmassen geschätzt, und ihr Durchmesser beträgt ungefähr 152.000 Lichtjahre.

Die Milchstraße [14] hat wiederum ungefähr eine Gesamtmasse von 1,2 Billionen Sonnenmassen und einen Durchmesser von 100.000 Lichtjahren.

Start Positionen der Galaxien Milchstraße und Andromeda Galaxie

Mit einer Hubble-Konstante [8] von beginnt der Verschmelzungsprozess der Beiden Galaxien in 3,1 Milliarden Jahren:



Nach 3,2 Milliarden Jahren, Mitte des Verschmelzungsprozesses

Nach 3,5 Milliarden Jahren, Galaxie stabilisiert sich und ist eine kugelförmige elliptische Galaxie

Nach 3,1 Milliarden Jahren, der Verschmelzungsprozess beginnt

Unsere Software zeigt realistisch wie nach der Verschmelzung der beiden Spiralgalaxien eine große elliptische Galaxie entsteht, da die Dichtewellen durch die Verschmelzung zerstört werden. Diese neue Galaxy wird Milkdromeda genannt.

# 13. Kollision der Whirlpool-Galaxie (M51) mit NGC 5195

# 14. Fazit und Ausblick

Wir konnten beweisen, dass unsere Algorithmen korrekte Simulationsdaten und Aussagen über die Zukunft mit einem bekannten maximalen Fehler treffen können.

Unsere Software kann Sonnensysteme, elliptische Galaxien und Spiralgalaxien mit beliebiger Anzahl von Objekten und mit korrekter Massenverteilung simulieren.

Außerdem können wir die Simulierten Daten dreidimensional grafisch darstellen. Der Beobachter kann sich dabei frei im Raum bewegen und kann die Zeit beliebig wählen.

Da unsere Software opensource ist und einfache Schnittstellen hat ist es für einen weniger erfahrenden Entwickler niederschwellig möglich den Funktionsumfang unserer Software zu erweitern.

# Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | N-body\_problem, „Wikipedia,“ [Online]. Available: https://en.wikipedia.org/wiki/N-body\_problem. [Zugriff am 28 12 2023]. |
| [2] | P. S. I. Charles Dyer, „Softening in N-Body Simulations of Collisionless Systems,“ Department of astronomy and Scarborough College, 1993. |
| [3] | V. Springel, „Smoothed Particle Hydrodynamics in Astrophysics,“ Max-Planck-Institut f¨ur Astrophysik, D-85741 Garching, 2011. |
| [4] | Semi-implicit\_Euler\_method, „Wikipedia,“ [Online]. Available: https://en.wikipedia.org/wiki/Semi-implicit\_Euler\_method. [Zugriff am 10 12 2023]. |
| [5] | Runge–Kutta\_methods, „Wikipedia,“ [Online]. Available: https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta\_methods. [Zugriff am 28 11 2023]. |
| [6] | Leapfrog integration, „Wikipedia,“ [Online]. Available: https://en.wikipedia.org/wiki/Leapfrog\_integration. [Zugriff am 8 12 2023]. |
| [7] | „NASA Horizons System,“ [Online]. Available: https://ssd.jpl.nasa.gov/horizons/. [Zugriff am 10 11 2023]. |
| [8] | Hubble-Konstante, „Wikipedia,“ [Online]. Available: https://de.wikipedia.org/wiki/Hubble-Konstante. [Zugriff am 6 11 2023]. |
| [9] | Barnes-Hut-Algorithmus, „Wikipedia,“ [Online]. Available: https://de.wikipedia.org/wiki/Barnes-Hut-Algorithmus. [Zugriff am 1 11 2023]. |
| [10] | Kimi Sickinger und Philip Späth, „Source Code auf Github,“ [Online]. Available: https://github.com/Philip-Spaeth/simulation. |
| [11] | Hubble-Sequenz, „Wikipedia,“ [Online]. Available: https://de.wikipedia.org/wiki/Hubble-Sequenz. [Zugriff am 12 12 2023]. |
| [12] | Navarro-Frenk-White profile, „Wikipedia,“ [Online]. Available: https://en.wikipedia.org/wiki/Navarro%E2%80%93Frenk%E2%80%93White\_profile. [Zugriff am 30 12 2023]. |
| [13] | Andromeda Galaxy, „Wikipedia,“ [Online]. Available: https://en.wikipedia.org/wiki/Andromeda\_Galaxy. [Zugriff am 31 12 2023]. |
| [14] | Milky Way, „Wikipedia,“ [Online]. Available: https://de.wikipedia.org/wiki/Milky\_Way. [Zugriff am 10 1 2024]. |

Unterstützungsleistungen

Sowohl die Idee als auch die gesamte Umsetzung ist allein unser Werk. Wir haben lediglich Unterstützung bei der Erklärung bestimmter Formeln und Konzepten wie zum Beispiel der Runge Kutta Methode, von unserem Projektbetreuer Kai Marquardt erhalten. Der gesamte Code wurde von uns geschrieben, KIs wurden lediglich dazu verwendet, um proprietäre Softwarebibliotheken (Beispiel OpenGL), in unser Projekt einzubinden.

1. [↑](#endnote-ref-2)