

Simulation von Gravitation im Sonnensystem

Till Wegener (28891)

Hochschule Rhein-Waal

Fakultät Kommunikation und Umwelt

Methoden und Werkzeuge der modernen Astronomie

Moers, Deutschland

till.wegener@hsrw.org

Abstract—Die Bewegung der Sterne und Planeten fasziniert die Menschheit schon seit Urzeiten. In diesem Paper wird eine Methode zur Simulation der Gravitation im Sonnensystem vorgestellt, welche physikalisch akkurate Daten verwendet, um die Bewegung der Planeten zu berechnen. Die Simulation wird in C++ implementiert und visualisiert.

Index terms—Simulation, Gravitation, Sonnensystem, C++, Visualisierung

CONTENTS

I. Einleitung	1
II. Historischer Kontext	1
III. Mathematische Grundlagen	1
IV. Vorbereitung	2
IV.A. Datensätze	2
IV.B. Wahl der Programmiersprache	2
IV.C. Wahl von Hilfsbibliotheken	2
V. Implementierung	2
VI. Probleme	3
VI.A. 32-Bit gleitkommazahlen	3
VI.B. Finden von passenden Datensätzen	3
VI.C. Bestimmen von Initialwerten	3
VII. Ergebnisse	3
VIII. Diskussion	3
Quellen	3

I. EINLEITUNG

Die Planeten in unserem Sonnensystem folgen wie alle anderen Objekt im Universum den Physikalischen Grundgesetzen. Für die Bewegungen der Planeten ist primär die Gravitationskraft verantwortlich. Durch das verstehen dieser Kraft ist es uns möglich, die Bewegungen der Planeten zu simulieren.

Dieses Paper verwendet diese Grundgesetze um eine Simulation zu entwickeln, welche die Bewegungen der Planeten im Sonnensystem berechnet.

II. HISTORISCHER KONTEXT

Im Laufe der Zeit haben viele Physiker und Astronomen sich mit der Bewegung der Planeten beschäftigt. Über die Jahrhunderte wurden viele Modelle entwickelt welche nah am heutigen Stand der Wissenschaft sind. Im folgenden werden einige dieser Modelle vorgestellt.

1. Im 16ten Jahrhundert entwickelte Nikolaus Kopernikus mit *Über die Umlaufbahnen der Himmelsphären* [1] das erste Modell, welches den heutigen Erkenntnissen nahe kommt. Er beschrieb die Bewegung der Planeten dadurch, dass alle Planeten aufeinander Gravitation ausüben.
2. Zu Beginn des 17ten Jahrhunderts entwickelte Johannes Kepler mit *Astronomia nova* [2] ein Modell, welches die Bewegung der Planeten genauer beschrieb. Seine Axiome besagen, dass die Planeten sich auf Ellipsen bewegen, und Körper sich gegenseitig anziehen.
3. 1644 entwickelte Isaac Newton mit *Philosophiæ Naturalis Principia Mathematica* [3] das Modell, welches bis heute zur Beschreibung der Gravitation verwendet wird. Seine Gravitationsgesetze beschreiben, wie sich Körper aufeinander anziehen.

III. MATHEMATISCHE GRUNDLAGEN

Wie durch Newton beschrieben, wirkt auf jeden Körper eine Gravitationskraft, welche durch die Masse der Körper und den Abstand zwischen den Körpern bestimmt wird. Die Gravitationskraft zwischen zwei Körpern kann durch die Formel

$$F_G = G * \frac{m_1 * m_2}{r^2} \quad (1)$$

beschrieben werden. Hierbei ist F_G die resultierende Gravitationskraft, G die Gravitationskonstante ($6.67 * 10^{-11} \frac{m^3}{kg*s^2}$), m_1 und m_2 die Massen der Körper und r der Abstand zwischen den Körpern. Die Kraft wirkt hierbei entlang der Verbindungslinie der Körper.

Für ein 2-Körper-System kann die Bewegung der Körper durch diese Formel bestimmt werden. Die Gravitationskraft

welche die Körper aufeinander ausüben, kann mit der Masse dieser verwendet werden, um die Beschleunigung der Körper mit $F = m * a$ zu bestimmen.

Für 2-Körper-Systeme ist es möglich, numerische Lösungen für die Bewegung der Körper zu finden.

In mehr-Körper-Systemen wird die Bewegung der Körper durch die Summe der einzelnen wirkenden Gravitationskräfte bestimmt. Für Systeme mit 3 oder mehr Körpern ist es nicht möglich, analytische Lösungen zu finden. Wenn wir hierzu Lösungen finden wollen, müssen wir numerische Methoden verwenden.

Im folgenden wird eine Methode vorgestellt, welche Eulers Methode verwendet, um die Bewegung der Planeten im Sonnensystem zu simulieren.

Eulers Methode ist eine numerische Methode, welche einen Differentialgleichungssystem durch diskrete Schritte approximiert. Die Methode ist einfach zu implementieren, kann jedoch aufgrund ihrer funktionsweise ungenau sein. Als erstes bestimmen wir unseren diskreten Zeitschritt Δt . Dannach bestimmen die Startwerte für die Position und Geschwindigkeit der Körper im Raum.

Nun können wir für alle n Körper die Gravitationskräfte berechnen, welche auf die Körper wirken. In diesem Fall ist die Kraft \vec{F}_G ein Vektor, welcher die Richtung und Stärke der Kraft angibt. Mit dieser Kraft können wir die Beschleunigung der Körper bestimmen. Mit $\vec{a} = \frac{\vec{F}_G}{m}$ und $\vec{v} = \vec{v} + \vec{a} * \Delta t$ können wir nun die angepassten Geschwindigkeiten der Körper bestimmen. Die Geschwindigkeit ist in diesem Fall ein 3-Dimensionaler Vektor, welcher die Geschwindigkeit in x , y und z Richtung angibt. Für einen Körper kann nun die neue Position \vec{r} im Raum mit $\vec{r} = \vec{r} + \vec{v} * \Delta t$ bestimmt werden.

Wenn wir dieses Verfahren nun für alle Körper im System durchführen, können wir die Bewegung der Körper simulieren. Dieses Verfahren von hand durchzuführen ist sehr aufwendig, weshalb sich die implementation als Computerprogramm anbietet.

IV. VORBEREITUNG

A. Datensätze

Für die Simulation des Sonnensystems benötigen wir Daten über die Planeten und ihre diversen Satelliten. Die wichtigen Daten für die Simulation sind die Masse der Körper, die Position der Körper im Raum und die Geschwindigkeit der Körper. Diese Daten können aus diversen Quellen bezogen werden. Für die Planeten wurden die Daten aus dem Projekt *Nasa Data Scraper* von *Devstronomy* [4] auf github verwendet. Das gleiche Projekt beinhaltet auch einen Datensatz für die Satelliten der Planeten. Dieser ist allerdings für unsere Simu-

lation nicht geeignet, da keine Informationen zu Position und Geschwindigkeit der Satelliten enthalten sind.

Die Daten der Satelliten wurden vom *Solar System Dynamics* Projekt des *Jet Propulsion Laboratory* der NASA [5] verwendet. Vom *SSD* Projekt können wir sowohl die Physikalischen Daten der Satelliten [6], als auch die orbitale Parameter der Satelliten [7] beziehen. Diese Daten beinhalten die *Semi Major Axis* und weitere Parameter, welche für die Simulation benötigt werden.

B. Wahl der Programmiersprache

Da für diese Simulation eine große Anzahl an Berechnungen durchgeführt werden muss, stellte die Wahl der Programmiersprache eine wichtige Entscheidung dar. Für diese Simulation wurde deshalb C++ gewählt. C++ ist eine schnelle kompilierte Sprache.

C. Wahl von Hilfsbibliotheken

Zur Visualisierung unserer Simulation benötigen wir eine Bibliothek, welche das darstellen von 3D-Objekten ermöglicht. Hierzu wurde die Bibliothek *Raylib* [8] verwendet. *Raylib* ist eine C-Bibliothek, welche das erstellen von 2D und 3D Anwendungen ermöglicht. Neben der Darstellung von 3D-Objekten war es auch noch wichtig, dass wir einfache Interface-Elemente anzeigen können. Hierzu wird die Bibliothek *Dear ImGui* [9] verwendet. *ImGui* ist eine C++ Bibliothek, welche in der Spieleentwicklung verwendet wird, um einfache Interface-Elemente anzuzeigen. Um diese beiden Bibliotheken zusammen zu verwenden, wurde das Projekt *rImGui* [10] verwendet.

Die Bibliotheken *ImGui* und *rImGui* wurden als Submodule in das Projekt eingebunden. *Raylib* wird während der kompilierung des Projektes heruntergeladen und verwendet. Hierzu wurde das Programm *Cmake* [11] verwendet. *Cmake* ist ein Build-System, welches es ermöglicht, C und C++ Projekte plattformunabhängig zu kompilieren.

V. IMPLEMENTIERUNG

Die Implementierung der Simulation folgt dem zuvor beschriebenen Ablauf. In der *Main*-Funktion der Simulation werden zuerst die nötigen Bibliotheken initialisiert. Für dieses Projekt war es nötig, einen eigenen *Vektor*-Datentypen zu verwenden. Die Gründe Hierzu werden in Section VI.A erläutert. Dieser *SciVec* verwendet 64-Bit gleitkomma-Zahlen, um möglichst präzise Berechnungen zu ermöglichen. Der Datentyp und die dazugehörigen Funktionen zur Addition, Subtraktion, Multiplikation und Skalierung können in *SciVec.h* gefunden werden. Die Körper des Sonnensystems werden mithilfe eines Datentyps *Body* abgebildet. In *Body.h* wird das *struct Body* wie folgt definiert:

```

struct Body
{
    std::string name;
    double radius;
    double mass;
    Color color{};
    SciVec3 position;
    SciVec3 velocity;
    bool isPlanet;

    int satelliteCount = 0;
    std::string satelliteNames[64] = {" "};

    ...
}

```

Die Felder *radius*, *mass*, *postion*, und *velocity* sind für die Simulation am wichtigsten. Für jedes Objekt wird außerdem ein *Name*, eine *Farbe*, und eine Liste an Satllieten gespeichert.

Die Datei *loader.h* beinhaltet die Logik, welche die Datensätze aus Section IV.A lädt.

VI. PROBLEME

- A. 32-Bit gleitkommazahlen
- B. Finden von passenden Datensätzen
- C. Bestimmen von Initialwerten

VII. ERGEBNISSE

VIII. DISKUSSION

QUELLEN

- [1] Nikolaus Kopernikus, *De revolutionibus orbium coelestium*. 1543.
- [2] *Astronomia nova*. 1609.
- [3] *Philosophiæ Naturalis Principia Mathematica*. 1644.
- [4] “NASA Data Scraper.” Jul. 08, 2024. [Online]. Available: <https://github.com/devstronomy/nasa-data-scraper/tree/master>
- [5] “Solar System Dynamics.” [Online]. Available: <https://ssd.jpl.nasa.gov/>
- [6] “Satellite physical data.” [Online]. Available: https://ssd.jpl.nasa.gov/sats/phys_par/
- [7] “Satellite orbital data.” [Online]. Available: <https://ssd.jpl.nasa.gov/sats/elem/>
- [8] raysan5, “Raylib.” Jul. 08, 2024. [Online]. Available: <https://github.com/raysan5/raylib.git>
- [9] ocornut, “ImGui.” Jul. 08, 2024. [Online]. Available: <https://github.com/ocornut/imgui.git>
- [10] “RImGui.” Jul. 08, 2024. [Online]. Available: <https://github.com/raylib-extras/rImGui.git>
- [11] “CMake.” [Online]. Available: <https://cmake.org/>