Simpert-Kraemer-Gymnasium **2006/2008**
Krumbach

Facharbeit

aus dem Fach

Physik

Thema: Computersimulation von geladenen Teilchen im elektrischen und magnetischen Feld

Verfasser:	Lochbrunner Matthias
Leistungskurs:	Physik
Kursleiter:	Std. Urban
Abgabetermin:	30. Dezember 2007

Erzielte Note: _____ In Worten: _____

Erzielte Punkte: _____ In Worten: _____
(einfache Wertung)

Abgabe im Sekretariat: _____

Unterschrift des Kursleiters: _____

Inhaltsverzeichnis

1 Einführung zum Thema

Zu Zeiten Isaac Newtons war die Physik noch ein sehr überschaubares Gebiet der Wissenschaft. Als Rechenhandwerk genügten primitivste Differenzial- und Integralrechnung vollkommen. Gleichungen dritten Grades waren eine Seltenheit. Und war mal ein neuartiges Phänomen gefunden, dass mit den bekannten Gesetzen nicht beschrieben werden konnte, bot sich immer noch die Möglichkeit die fehlende Erkenntisse durch ein Experiment zu erlangen. Der Einfallsreichtum unter den Versuchsaufbauten war oftmals erstaunlich. Doch spätestens seit dem ersten Versuch, einen Menschen auf den Mond zu schießen, wäre es ein recht teueres Unterfangen gewesen eine Vielzahl von Raketen gen Himmel zu schicken nur um herauszufinden, wie stark die Triebwerksköpfe sein müssten um das Vehikel zur gewünschten Umlaufbahn zu befördern. Hier wurde dann auch erstmals das Berechnen von Koordinaten und Kreisbahnen mit Papier und Bleistift ein Ding der Unmöglichkeit. Nicht nur weil die Rechnungen eines einzelnen Teilabschnittes unermäßlich lang und aufwändig wurde, sondern auch weil niemand die Verantwortung übernehmen wollte, wegen einem Rechenfehler ein Menschenleben zu riskieren. Ein neuer Weg musste gefunden werden, solch komplexe Berechungen schnell und exakt durchführen zu können. Man mag die Wissenschaftler wohl noch ausgelacht haben, als sie mit Lochstreifen bewaffnet den ersten Röhrenrießen das multiplizieren beibrachten um ihnen nach monatelangem gut Zureden dazu zu bringen, die Kurven möglicher Flugbahnen auszuspucken. Diese Hilfe verwendet man heute auch "'in der Elementarteilchenphysik, in der Materialforschung, Strömungsdynamik, Strukturmechanik, Chemie, Geo- und Astrophysik sowie Klima- und Umweltforschung"'[1]. Dazu reichen heutzutage allerdings keine Triodenrechner mehr, die in einer Sekunde eine Handvoll Rechenschritte durchführen, sondern es werden meist Großrechner wie der Leibnizrechner in Garching benötigt, um die rießige Datenflut und die überaus komplizierten Berechnungen, die dennoch nur Annäherungen sind, zu bewältigen.

Aber auch im Kleinen lassen sich durch Computersimulationen viele physikalische Phänomen anschaulich erklären. Eine große Hilfe für Physiklehrer, die im Unterricht bei Schülern oft auf Unverständis und mangelnder Vorstellungskraft treffen. Die Vorteile des Computereinsatzes im Unterricht liegen klar auf der Hand: So lassen sich durch eine geeignete Simulation das komplexe Zusammenspiel vieler Umwelteinflüsse auf einzellne wichtige Faktoren begrenzen und diese komplizierten Sachverhalte durch transpatente Modelle durchleuchten, da unnötige Einflüsse ausgeblendet werden. Idealbedingungen können geschaffen werden. Auch ermöglichen diese das beobachten physiklaischer Experimente die real im Unterrricht niemals durchführbar wären, wie zum Beispiel sehr langsame bzw. sehr schnell ablaufende Effekte wie

[1] Quelle : http://www.stmwfk.bayern.de/pressearchiv/meldung.asp?NewsID=649

ein elektischer Blitz, ebenso Versuche mit strahlenden oder hoch giftigen Materialien wären im Klassenzimmer unverantwortlich mit einem Computersimualtion jedoch anschaulich durchführbar[2]. Zwar bietet das Internet ein reichhaltiges Angebot physikalischer Simualtionsprogramme, oft sogar als Java-Applet oder Flash-Skript direkt im Internetbrowser ausführbar, doch nur die Wenigsten von ihnen arbeiten mit der dritten Dimension, und alle haben nur einen beschränkten Anwendungsbereich. Da sie meist nur geskriptet sind, bekommt der Betrachter quasi nur einen Film vorgesetzt, bei dem er manche Werte mit Reglern verstellen kann. Diese Nische zu füllen entschied ich mich, selber im Rahmen dieser Facharbeit, ein Programm zu entwickeln, das auf der Basis von Microsoft DirectX™geschrieben in *C++*, phyikalische Phänomene in den Bereichen Elektrostatik und Magnetismus im dreidimensionalem Raum simulieren kann. Hierfür setze ich auf das bewerte Editor-Prinzip, welches dem Anwender erlaubt sich aus einer Reihe von Bausteinen seine gewünschte Szene zusammen zu stellen, genauso wie es in komerziellen 3D-Simulationen wie Maya oder Cinema3D üblich ist. Somit lassen sich dann die Verhaltensmuster Elementarteichen in bestimmten Umgebungen beobachten; genauere Phänomene werden am Schluss dieser Arbeit behandelt. Zunächst werde ich kurz die Handhabung und den Umfang des Programms vorstellen, danach noch auf die physikalischen Gesetze eingehen, die in dieser Simulation Anwendung finden und wie sie im Code umgesetzt wurden. Die Angabe genauer Ergebisse und realitätsgetreuer Werte, die genau berechnete Konstanten im Quellcode vorraussetzt, ist nicht Aufgabe dieser Facharbeit. Und somit handelt es sich um eine reine Simulation ohne Messmöglichkeit.

[2]Quelle : http://www.medien.ifi.lmu.de/fileadmin/mimuc/mll_ws0506/Vortrag_Kim.pdf

2 Kurze Bedienungsanleitung

2.1 Installation

Vor der Installation sollte sichergestellt sein, dass auf dem PC ein lauffähiges "'Windows XP"' mit Service Pack 2 installiert ist und eine aktuelle Version der Hardwareschnittstelle "'DirectX 9.0c"' oder neuer bereits Verwendung findet. Da die Simulation bereits mit dem "'DirectX SDK"' von 2007 geschrieben wurde, empfiehlt es sich gegeben falls, die "'Runtime"' Version auf dem PC zu aktualisieren [3].
Nach Einlegen der CD in das Laufwerk erscheint automatisch ein mit "'Install Creater"' erzeugtes Installationsfenster. Sollte das Fenster nicht automatisch erscheinen, kann das Setup auf der CD auch manuell unter */Software/Setup.exe* gestartet werden. Die Anweisungen des Installationsassistenten erklären sich selbst. Nach erfolgreicher Installation erhält das Programm nun im *Startmenü*, unter *alle Progamme* einen Eintrag. Ebenfalls sind auf der CD noch weitere Dateien bezüglich dieser Facharbeit zu finden. So z.B. dieses Skript als pdf-Datei, sowie der Quellcode des Programms und noch zusätzliche Materialien. Gegebenenfalls muss dazu die CD im *Arbeitsplatz* mit Rechtsklick, *Öffnen*, da bei einem Doppelkick der *autorun* der CD das Installationsprogramm starten würde.

2.2 Freie Kamerabewegung

Das Gedrückthalten der *Alt*-Taste signalisiert dem Programm, dass die Maus nun zur Steuerung der Kamera verwendet wird. Jetzt lässt sich mit der mittleren Maustaste die Szene parallel zur Sichtebene verschieben, wobei der Cursor ein anderes Symbol erhält, eines mit einem Pfeil in jede Himmelsrichtung. Um sich im Raum zu drehen wird die linke Maustaste verwendet. Jede Drehung erfolgt immer um einen bestimmten Mittelpunkt. Dieser ist entweder die Position des zuletzt markierten Elements in der Szene, oder falls nichts markiert wurde, der Ursprung im Koordinatensystem. Als Cursor erscheinen nun zwei im Kreis laufende Pfeile. Der Zoom bzw. der Abstand der Kamera zur Szene wird mit der rechten Maustaste verändert. Hier gilt: Wird die Maus nach links oben verschoben, wird aus der Szene herausgezoomt, beim Verschieben nach rechts unten hineingezoomt. Allerdings ist zu beachten, dass der Zoom nicht linear zur Mausbewegung berechnet wird, sondern exponential. So wird ein für die meisten Situationen praktischeres und gleichsam harmonisches Zoomen ermöglicht. Des Öfteren kann es passieren, dass man sich irgendwo im Raum verliert und keine Orientierung mehr hat. Hierfür ist der Hotkey *F* gedacht, welcher die Kamera auf das zuletzt markierte Element zentriert bzw. auf den Ursprung.

[3] Download : http://www.microsoft.com/downloads/details.aspx?displaylang=de&FamilyID=2da43d38-db71-4c1b-bc6a-9b6652cd92a3

2.3 Arbeiten im dreidimensionalen Raum

In der Simulation stehen zwei Werkzeuge (englisch: "'Tools"') zur Verfügung: Eines um markierte Elemente zu Verschieben und ein Anderes um bestimmte Objekte, wie Kugeln und Kondensatorplatten , zu skalieren. Bei beiden Werkzeugen gilt stets die "'Dreifarbenregel"', die auch bei vielen anderen Animationsprogrammen Anwendung findet. Die drei Koordinatenachsen werden durch die drei Grundfarben repräsentiert: Rot für die X-Achse, Grün für die Y-Achse und Blau für die Z-Achse. Mit dem Hotkey *W* wird zu dem Verschiebungswerkzeug gewechselt, mit dem Hotkey *E* zum Skalierwerkzeug. Durch erneutes Drücken der jeweiligen Taste wird das Werkzeug ausgeblendet. Das Skalierwerkzeug erkennt man an den drei Pfeilen, die immer in die positive Richtung zeigen. Elemente werden verschoben, indem man die Pfeilspitzen des Werkzeuges mit der linken Maustaste anklickt und die Maus dann mit gedrückter Taste in die gewünschte Richtung schiebt. Oft ist es notwendig, die Szene aus dem richtigen Blickwinkel zu betrachten, um leichter arbeiten zu können. Äquivalent verhält es sich mit dem Skalierwerkzeug, zu erkennen an den drei Würfeln. Der Betrag der Veränderungen entlang der Koordinatenachsen wird mit Hilfe eines trigometrischen Ansatzes berechnet, wodurch es nach längerer Zeit leider zu Unregelmäßigkeiten kommen kann. Auch die wechselhafte Entfernung zur Kamera tut ihr Übriges dazu.

2.4 Arbeiten mit der "'Channelbox"'

Um genaue Angaben zu einem Objekt machen zu können, wurde der Simulation ein Kontrollfenster, oder im Fachchargon auch "'Channelbox"' genannt, hinzugefügt, mit dem der Benutzer in der Lage ist jeden Wert des markierten Elements genau angeben zu können. Das Kontrollfenster lässt sich im *Menü* unter dem Menüpunkt*Ansicht*, *weitere Fenster*, *Kontrollfenster* starten. Meist ist es jedoch schon zu Beginn zu sehen. Zu Beachten ist allerdings, dass nur manche Eigenschaften auf alle markierten Elemente gesetzt werden können. Angaben zu genauen Werten von z.B. Position oder Geschwindigkeit werden nur vom zuletzt markierten Element übernommen, da dieses in der Simulation einen besonderen Fokus erhält. Ebenfalls können von der *Channelbox* stets die genauen Werte abgelesen werden, die in einer ähnlichen Form, wie in den *.sim-Datein dargestellt sind. Dateien mit der Endung .sim werden dazu benutzt eine Szene abzuspeichern, welche zu einem späteren Zeitpunkt dann wieder geladen werden kann. Aufgrund technischer Schwierigkeiten ist es bereits noch nicht gelungen, diese Dateien direkt mit dem Programm öffnen zu lassen. Aber dennoch eignet sich dieses Dateiformat ausgezeichnet, es mit dem Microsoft Text-Editor oder irgendeinem anderen Text-Editor zu öffnen, um damit den Inhalt zu verändern. So lassen sich beispielsweise ohne das Starten der Anwendung

neue Elemente erstellen und sie mit den gewünschten Werten versehen.

2.5 Szene animieren

Dem eigentlichen Zweck als Simulation genüge zu tun und der 3D-Szene Leben einzuhauchen, kann mithilfe der Funktionstaste *F2* die Animation gestartet werden, in der dann automatisch alle physikalischen Berechungen durchgeführt werden. Durch erneutes Drücken der Funktionstaste wird die Animation wieder gestoppt. Mit der *TAB*-Taste wird erreicht, dass nur das nächste Bild, auf englisch : "'Frame"', berechnet und angezeigt wird. So lässt sich die Animation schrittweise abspielen, wobei der Benutzer in komplexen Szenen meist besser in der Lage ist, den Überblick zu behalten.
Eine weitere Erleichterung ist mit der Funktion, die Bahn eines Teilchens anzeigen zu lassen, gegeben. Somit lässt sich das Verhalten des Spurauslegers genauer analysieren und der Bertachter bekommt ein aussagekräftiges Bild von der Szene.

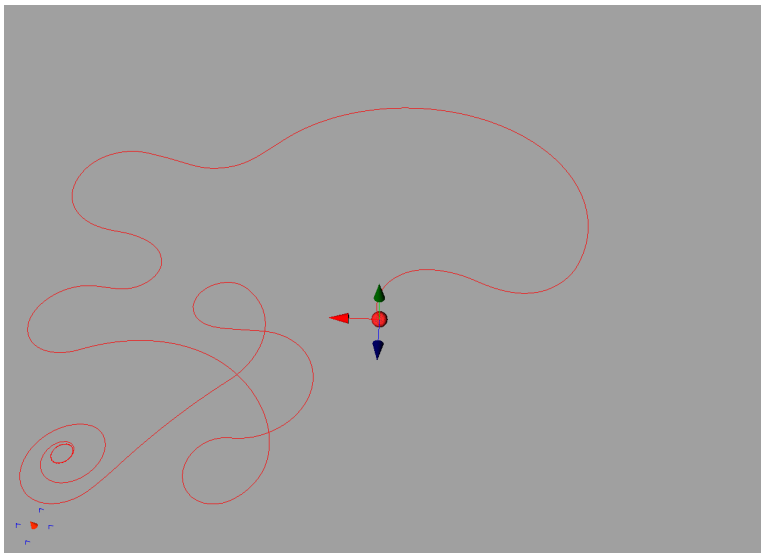

Abbildung 1: Spur eines Elektrons

3 Eingebettete physikalische Gesetze

3.1 Elektrostatik

3.1.1 Coulombsche Gesetz

Setzt man ein geladenes Teilchen in ein elektisches Feld, z.B. in das eines Plattenkondensators, so erfährt es dort eine Kraftauswirkung, das abhängig vom Vorzeichen der Ladung, mit oder entgegengesetzt des Feldes gerichtet ist($F = q \cdot E$). Da aber auch geladene Körper selbst ein elektrische Feld erzeugen, stehen zwei geladene Teilchen

stets in einer elektrostatischen Abhängigkeit. Man nennt diese das *Coulombsche Gesetz*.

Coulomb-Gesetz für punktförmige Ladung[4] :

$$\vec{F} = \frac{Q_1 \cdot Q_2}{4\pi \cdot \epsilon_0} \cdot \frac{1}{r^3} \cdot \vec{r}$$

Erläuterungen :

Q_1	:	Ladung des ersten Teilchens
Q_2	:	Ladung des zweiten Teilchens
$\vec{r}$	:	Entfernung der beiden Teilchen
r	:	Betrag von $\vec{r}$

3.1.2 Programmiertechnische Umsetzung

Bevor nun das *Gesetz von Coulomb* im Quellcode angewandt werden kann, muss zuerst der Radius, hier die Entfenung der beiden Teilchen, mithilfe des Satzes von Pythagoras ermittelt werden. Danach lässt sich die Formel trivial anwenden :

Satz von Pythagoras	:	d	=	$\sqrt{a^2 + b^2 + c^2}$
Daraus folgt für den Betrag von $\vec{r}$	:	$\lvert\vec{r}\rvert$	=	$\sqrt{\vec{r_x}^2 + \vec{r_y}^2 + \vec{r_z}^2}$
Eingestzt in das Coulombsche Gesetz	:	$\vec{F}$	=	$Konstante \cdot \vec{r} \div \lvert\vec{r}\rvert^3$

Bemerkung : Da in der gesamten Simulation keinerlei Wert auf Maßstabstreue gelegt wird, fällt die *Konstante* im Quellcode schlicht weg.

Das Ganze zu einer Funktion namens *ForceElectrical(...)* zusammengefasst:

Listing 1: Quellcodeausschnitt aus Physics.cpp, Zeile 798 - 807

```
1  D3DXVECTOR3 CPhysics::ForceElectrical(D3DXVECTOR3 DrivenPosition, D3DXVECTOR3 DriverPosition){
2          D3DXVECTOR3 v;
3          float Distance = sqrt((DrivenPosition.x - DriverPosition.x) * (DrivenPosition.x -
               DriverPosition.x) +
4                  (DrivenPosition.y - DriverPosition.y) * (DrivenPosition.y - DriverPosition.y) +
5                  (DrivenPosition.z - DriverPosition.z) * (DrivenPosition.z - DriverPosition.z));
6
7          v = (DrivenPosition - DriverPosition) / (Distance * Distance * Distance);
8
9          return v;
10 }
```

Erläuterungen :

[4]Quelle : Physik, Leistugskurs 1. Semester, Seite 61

DrivenPosition	:	Position des Teilchens, auf das die *Coulombkraft* wirkt
DriverPosition	:	Position des Teilchens, das die *Coulombkraft* hervorruft
Distance	:	Entfernung der beiden Teichen ($\vert\vec{r}\vert$)

3.1.3 Problem des quadratischen Abstandgesetzes

Soweit die Theorie. Leider tritt bei dieser Konstellation nicht selten das Phänomen auf, dass ein Elektron, das soeben noch ein Proton umkreist hat, plötzlich katapultartig in den Weiten des Raumes verschwindet, ohne dass dies irgendeiner physikalischen Gesetzmäßigkeit folgt. Dies passiert ganau dann, wenn sich ein Elektron einem Proton so sehr genähert hat, dass bei der Berechnung der Beschleunigungskraft der Wert so groß ist, dass dieser das Elektron beim nächsten Rechenschritt derart weit vom Proton entfernt, dass die jetzige Kraftwirkung, die die Elementarteilchen wieder vereinen sollte einen vernachlässigbar kleinen Wert annimmt.

3.2 Magnetisches Feld

3.2.1 Physikalischer Hintergrund

"'Bewegen sich positive Ladungsträger senkrecht zu den Magnetfeldlinien, so erfahren sie eine Kraft $\vec{F_1}$, die sich mit der Drei-Finger-Regel der rechten Hand bestimmen läßt "' [5]. Natürlich gilt dies auch für negative Ladungsträger, wenn man die linke Hand zu Rat zieht. Man nennt diese Kraft *Lorenzkraft*, mit der allgemeinen Gleichung:

$$F = q \cdot v \cdot B$$

Bemerkung : Alle drei Vektoren, $\vec{F}$, $\vec{v}$ und $\vec{B}$, stehen jeweils senkrecht zueinander.

3.2.2 Programmiertechnische Umsetzung

Um einen Wirkungsvektor aus den gegebenen Größen berechnen zu können, müssen zunächst alle Angaben in ihre Komponenten aufgeteilt werden und dann jede zu berechnende Wirkungskomponente aus dem Produkt der entsprechenden Geschwindigkeits- und Feldkomponente ermittlet werden.

[5] Quelle : Physik, Leistugskurs 1. Semester, Seite 110/111

$$
\begin{aligned}
&\text{Lorenzkraft} \quad : \quad F = q \cdot v \cdot B \\
&\text{Daraus folgt} \quad : \quad F_{x1} = q \cdot v_y \cdot B_z \\
&\qquad\qquad\quad\;\; F_{x2} = q \cdot v_z \cdot B_y \\
&\qquad\qquad\quad\;\; F_{y1} = q \cdot v_x \cdot B_z \\
&\qquad\qquad\quad\;\; F_{y2} = q \cdot v_z \cdot B_x \\
&\qquad\qquad\quad\;\; F_{z1} = q \cdot v_x \cdot B_y \\
&\qquad\qquad\quad\;\; F_{z2} = q \cdot v_y \cdot B_x
\end{aligned}
$$

Die somit ermittelten 6 Gleichungen, jeweils zwei für jede Koordinatenachse, lassen sich nun zu einem Kraft- bzw. Beschleunigungsvektor addieren. Allerdings wurde bis jetzt noch kein Blick auf die Vorzeichen der einzellnen Teilergebnisse geworfen. Diese lassen sich jedoch mithilfe der "'Dreifingerregel"' relativ leicht bestimmen.

Listing 2: Quellcodeausschnitt aus PhysicsOld.cpp, Zeile 768 - 781

```
1  D3DXVECTOR3 CPhysics::ForceMagnetical(D3DXVECTOR3 DrivenSpeed, D3DXVECTOR3 DriverForce){
2          D3DXVECTOR3 v = D3DXVECTOR3(0.0f, 0.0f, 0.0f);
3
4          v.x += -DrivenSpeed.z * DriverForce.y;
5          v.z += +DrivenSpeed.x * DriverForce.y;
6
7          v.y += +DrivenSpeed.z * DriverForce.x;
8          v.z += -DrivenSpeed.y * DriverForce.x;
9
10         v.x += +DrivenSpeed.y * DriverForce.z;
11         v.y += -DrivenSpeed.x * DriverForce.z;
12
13         return v;
14 }
```

Bemerkung : Der Faktor q wird außerhalb der Funktion behandelt.

3.2.3 Problem der endlichen Zeiteinteilung

Leider scheint diese Rechung nicht ganz die Realität zu beschreiben, denn bei der Ausführung dieses Codes werden die Teilchen im Magnetfeld stets schneller, wohingegen kein Grund dazu besteht. Vielmehr wiederspricht dieses Phänomen sogar dem Energieerhaltungssatz von *Newton*. Bei genauerer Betrachtung des Rechenmodells fällt dabei folgendes auf:
Rechnerisch Zusammengefasst :

$$
\begin{aligned}
&\text{Satz des Pythagoras} \quad : \quad c = \sqrt{a^2 + b^2} \\
&\text{Daraus folgt für } b \neq 0 \quad : \quad c > a
\end{aligned}
$$

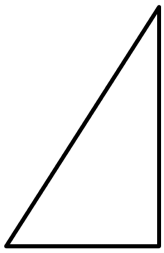

Abbildung 2: Pythagoras der magnetischen Beschleunigung

Laut dieser Gleichung müsste die Geschwindigkeit immer größer werden; zwar nicht in besonders großem Maße, aber dennoch mit der Zeit deutlich spürbar. In der Natur hat b jedoch keinen endlichen Wert, sondern einen infinitisimal kleinen , wodurch sich c und damit auch der Betrag der Geschwindigkeit nicht verändert. Dieser unerwünschte Effekt lässt sich durch einen kleinen Eingriff in der Berechung nahezu eliminieren. Hier kommt die Konstante *MAGNETICAL_HOLD* ins Spiel. Mit ihr wird versucht der fälschlich eingeschlichenen Beschleunigung entgegen zu wirken. Der Wert der Größe wurde experimental ermittelt und unterliegt dadurch auch einer gewissen Ungenauigkeit. Genauere Untersuchungen, die Erläuterung dazu würde den Rahmen dieser Facharbeit sprengen, ergeben, dass die Abweichung vom Quadrat der Felstärke abhängt. Nach weiteren Umformungen ergibt sich dann daraus folgende Funktion:

Listing 3: Quellcodeausschnitt aus Physics.cpp, Zeile 810 - 835

```
1  D3DXVECTOR3 CPhysics::ForceMagnetical(D3DXVECTOR3 DrivenSpeed, D3DXVECTOR3 DriverForce){
2          D3DXVECTOR3 v = D3DXVECTOR3(0.0f, 0.0f, 0.0f);
3
4          v.x += -DrivenSpeed.z * DriverForce.y;
5          v.z += +DrivenSpeed.x * DriverForce.y;
6
7          v.y += +DrivenSpeed.z * DriverForce.x;
8          v.z += -DrivenSpeed.y * DriverForce.x;
9
10         v.x += +DrivenSpeed.y * DriverForce.z;
11         v.y += -DrivenSpeed.x * DriverForce.z;
12
13         v.x += DrivenSpeed.x * DriverForce.y * DriverForce.y * MAGNETICAL_HOLD;
14         v.x += DrivenSpeed.x * DriverForce.z * DriverForce.z * MAGNETICAL_HOLD;
15         v.x += DrivenSpeed.x * DriverForce.x * DriverForce.x * MAGNETICAL_HOLD;
16
17         v.y += DrivenSpeed.y * DriverForce.x * DriverForce.x * MAGNETICAL_HOLD;
18         v.y += DrivenSpeed.y * DriverForce.z * DriverForce.z * MAGNETICAL_HOLD;
19         v.y += DrivenSpeed.y * DriverForce.y * DriverForce.y * MAGNETICAL_HOLD;
20
21         v.z += DrivenSpeed.z * DriverForce.y * DriverForce.y * MAGNETICAL_HOLD;
22         v.z += DrivenSpeed.z * DriverForce.x * DriverForce.x * MAGNETICAL_HOLD;
23         v.z += DrivenSpeed.z * DriverForce.z * DriverForce.z * MAGNETICAL_HOLD;
24
25         return v;
26 }
```

4 Objekte

4.1 Elementarteilchen

Die Grundlage dieser Simulation dürften wohl die Elementarteilchen sein. Mit ihnen werden nehezu alle Versuche durchgeführt. Das sind die positiv geladenen Elektronen so wie das negative Gegenstück das Elektron. Da wie oben schon erwähnt in diesem Programm keine der wirklichkeit entsprechenden Maßstäbe zum Einsatz kommen, genügt hier die Annäherung in der das Proton die 1000-fache Masse des Elektrons besitzt. Dies entspricht sogar in etwa dem natürlichen Wert. Diese Objekte sind in der Simulation unter *Hinzufügen* $->$ *Mikroteilchen* $->$ *Proton* bzw. *Elektonen* zu finden. Sollte mal einmal die Elementarteilchen für einen Versuch ungeeignet sein, so besteht noch die Möglichkeit einen Versuchskörper zu erstellen, bei dem alle Werte manuell anzugeben sind. Vergleichber mit einer Metallkugel sind hier zusätzlich Angaben zur Größe und Ladung möglich. Zu finden unter *Hinzufügen* $->$ *Makroteilchen* $->$ *Metallkugel* . Bei disem Element kann nun auch das oben genannte *Skalierwerkzeug* angewandt werden.

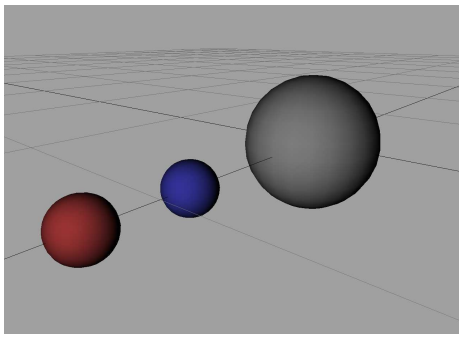

Abbildung 3: Elementarteilchen

4.2 Homogene Felder

Um dem Aufrag gerecht zu werden und eine "'Computersimulation geladerner Teilchen in elektrischen und magnetischen Feldern"' zu erhalten, befinden sich natürlich auch homogene Felder im Reportua dieses Programms. Alle zu finden unter *Hinzufügen* $->$ *Homogene Felder*. Hier stehen dem Benutzer drei Feldarten zur Auswahl: Das elektrische Feld, das magnetische Feld und das Gravitationsfeld. Aufgrund ihrer homogenen Eigenschaft können diese überall im Raum versetzt werden, ohne das es irgendeinen Einfluss auf das Geschehen nimmt. Wesentlich ist nur der Feldvektor, der mithilfe der *Channelbox* eingeblendet werden kann. Dieser lässt sich wahlweise ebenfalls mit der *Channelbox* oder mit dem *Verschiebewerkzeug* zurecht drehen. Auch das Drehen und Sklalieren des Wirkungsvektors eines Gravitationfeldes ist möglich. Mit ihm lassen sich z.B. Perabelbahnen des freien Falls sehr schön

darstellen. Zusammenfassend sind die homogenen Felder einfache Elemente, die ein relativ komplexeres System von z.B. Kondensatorplatten oder Helmholzspulenpaare stellvertretend ersetzen.

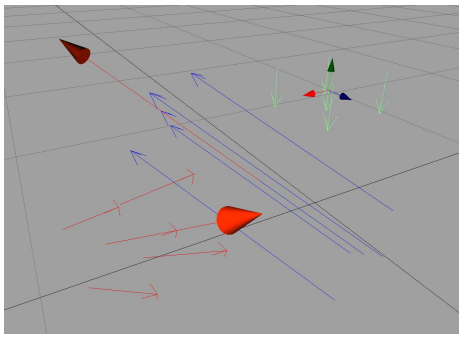

Abbildung 4: Homogene Felder

4.3 Kondensatorplatte

Da homogene Felder für manche Versuche nicht ausreichen könnten, weil sie z.B. einen unbegrenzten Wirkungsereich besitzen, kann das elektrische Feld auch durch zwei Kondensatorplatten ersetzt werden, welche ein inhomogenes Feld erzeugen. Das Verwenden von Kondensatorplatten ist allerdings ungleich komplizierter: Hier muss zunächst die Anzahl der Aufteilungen bestimmt werden. Dies ist wichtig, damit das Programm weiß, mit wie vielen "'Ladungseinheiten"' es rechnen soll. Da ein Simulation von Milliarden positiver bzw. negativer Ladungen ein Ding der Unmöglichkeit wär, beschrenkt sich dieses Programm auf eine geringere Anzahl, die der Benutzer selber angeben sollte. Die als Standartwert angegebenen zwei Aufteilungen eignen sich nicht besonders gut um das elektrische Feld eines Kondensators realistisch zu simulieren. Der Wert der hier zu wählen ist hängt von der Anzahl der Elementarteilchen ab, die sich in der Szene befinden und von der CPU-Leistung des Rechners, da dies bei Werten im dreistelligen Bereich sehr rechenaufwändig ist. Je höher der gewählte Wert, desto genauer das inhomogene Feld und desto höher sómit der Realitätgrad. Sobald eine neue Anzahl in der *Channelbox* eingegeben wurde, beginnt das Programm ein naturgemäße Verteilung der Elementarladungen. Dieser Vorgang kann dann durchaus 15 Minuten dauern und sollte nach dem endgültigen Skalieren der Platte manuell wiederholt werden.

4.4 "'Emitter"'

Der Emitter (lat. emittere = aussenden; ausstoßen) stellt eine flexible Elektronenkanone der Szene dar, welche dazu verwendet werden kann, andauernd Elektronen auszusenden. Der Vorteil dieses Instruments gegenüber seines realen Gegenstücks

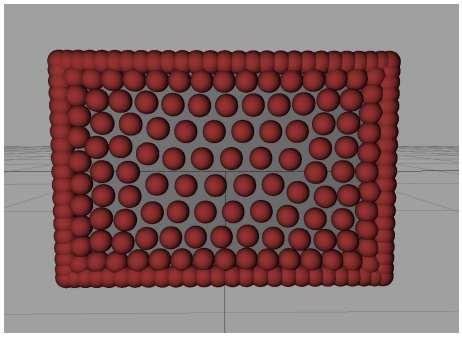

Abbildung 5: Kondensatorplatte

liegt in der Vielzahl der Einstellungsmöglichkeiten. Hier lassen sich neben der Position des Geräts auch Richtung und Betrag der auszustoßenden Teilchen angeben, und um wieviel von diesem Wert abgewichen werden darf. Da eine Elektronenkanone, bei der jedes Elektron genau den selben Geschwindigkeitsvektor bekäme, recht ausdrucksschwach wäre, weicht hier jedes Teilchen in der Startgeschwindigkeit um einen pseudozufälligen Vektor ab, der in der *Channelbox* unter den Feldern *Zufall X, Zufall Y und Zufall Z* indivieduell skaliert werden kann. Soll in einer Richtung keine Abweichung stattfinden ist hier der Wert Null einzugeben. Auch der zeitliche Abstand mit der die Ladungsträger das Gerät verlassen lässt sich manuell vorherbestimmen. Die Einheit des Abstand ist allerdings "'Frames[6]"' und nicht Sekunden oder sonst eine feste Zeiteinheit.

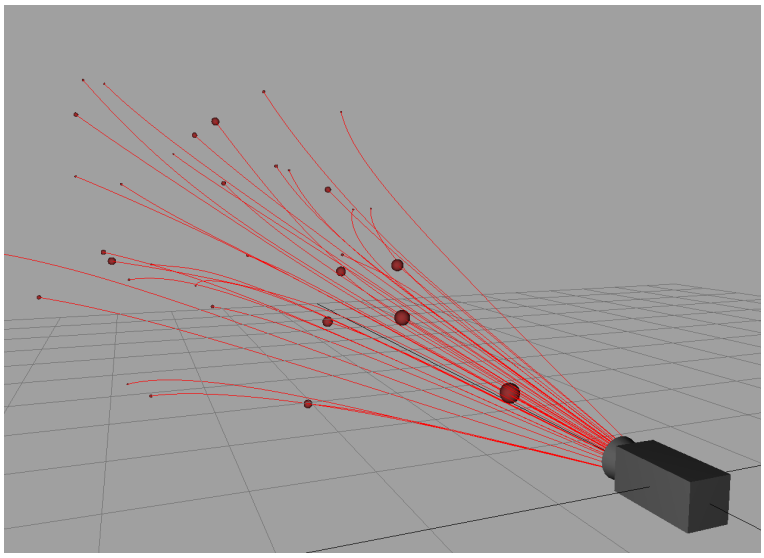

Abbildung 6: Emitter

[6]Bilder die am PC dargestellt werden

5 Halbschrittverfahren

5.1 Verfahren von Heun

Ende des 19. Jahrhunderts beschäftigte sich der Mathematiker und Philosoph Karl Heun mit der Berechung "'lnearer Differnzialgleichungen zweiter Ordnung, deren Lösungen durch den Kettenbruchalgortihmus verknüpft sind"'[7]. Unter anderm beschrieb er dort auch einen Therm der es ermöglicht, eine relativ gute Annäherung der Stammfunktion zu erhalten, wenn der Funktionstherm nur durch einzelne Werte beschrieben werden kann. Daher eignet er sich zum Beispiel gut bei der Berechnung von Planetenbahnen oder aber auch zur Berechnung Elementarteilchen in verschiedenen Feldern. Hier ist prinzipiell nur die Kraftwirkung gegeben, aus der dann über den Weg der Geschwindigkeit die Position des Teilchens bestimmt werden muss. Nach dem Hauptsatz der Integralrechnung ist also die Geschwindigkeit das Integral der Beschleunigung und die Position das Integral der Geschwindigkeit. Da die Beschleunigung jedoch nicht kontinuierlich berechnet wird, sondern nur alle X Millisekunden, läuft man leicht Gefahr, dass die daraus berechneten Integrale stark von der Wirklichkeit abweichen. Diese Gefahr lässt sich mit dem Verfahren von Heun zwar nicht vollständig auslöschen, aber dennoch stark minimieren.

Es sei gegeben die Funktion	:	$f(x)$	$=$	$m \cdot x + t$
Auf Grund der Linearität	:	$f(x_1 + \Delta x)$	$=$	$f(x_1) + f'(x_1) \cdot \Delta x$
Daraus folgt nach Heun	:	$f(x_1 + \Delta x)$	$=$	$f(x_1) + \frac{f'(x_1)+f'(x_1+\Delta x)}{2} \cdot \Delta x$

Was für die lineare Funktion exakt gilt, wendet Heun als Näherung auch bei nicht linearen Funktionen an. Dieses Verfahren hat sich in der Praxis sehr bewährt und findet daher häufig bei Simulationen Anwendung, die keinen vollständigen Funktionstherm besitzen [8].

Am Rande bemerkt: Carl Runge und Martin Wilhelm Kutta erweiterten das Verfahren von Heun, indem sie noch zusätzlich in einem ganzen Schritt mehrere sogenannte "'Stützpunkte"' verteilten und die Ableitungen an diesen dann zur Berechnung des Mittels einfließen ließen.

[7] Quelle : http://www-history.mcs.st-andrews.ac.uk/Biographies/Heun.html
[8] Quelle (von 29.12.2007): http://www.acdca.ac.at/material/kl8/numerik.pdf

$$\text{Heun} \quad : f(x_1 + \Delta x) = f(x_1) + \frac{f'(x_1)+f'(x_1+\Delta x)}{2} \cdot \Delta x$$

$$\text{Erweitert} \quad : f(x_1 + \Delta x) = f(x_1) + \frac{f'(x_1)+2\cdot f'(x_1+\frac{1}{3}\cdot\Delta x)+2\cdot f'(x_1+\frac{2}{3}\cdot\Delta x)+f'(x_1+\Delta x)}{6} \cdot \Delta x$$

$$\text{Allgemein} \quad : f(x_1 + \Delta x) = f(x_1) + \frac{f'(x_1)+2\cdot f'(x_1+\frac{1}{n}\cdot\Delta x)+\ldots+2\cdot f'(x_1+\frac{n-1}{n}\cdot\Delta x)+f'(x_1+\Delta x)}{2\cdot n} \cdot \Delta x$$

$$\text{Kurz} \quad : f(x_1 + \Delta x) = f(x_1) + \frac{\Delta x}{2\cdot n} \cdot \left(\sum_{i=0}^{n-1} f'(x_1 + \frac{i}{n} \cdot \Delta x) + \sum_{i=1}^{n} f'(x_1 + \frac{i}{n} \cdot \Delta x)\right)$$

Bemerkung : Die Zahl n ($n \in$) gibt die Anzahl der gesetzen Stütpunkte an.

Da allerdings das Setzen von Stützpunkten erneut eine Menge Rechenleistung des CPUs verbrauchen würde, wird dieses Verfahren in dieser Simulation nicht angewendet.

5.2 Programiertechnische Anwendung

Die Programiertechnische Anwendung gestaltet sich denkbar einfach: Man addiert zuerst die Hälfte der Ableitung multipliziert mit einer Konstante, berechnet dann die neuen Beschleunigungsvektoren und addiert zum Schluss den zweiten Teil der Ableitungen:

Listing 4: Quellcodeausschnitt aus einer bearbeitete Version von Physics.cpp, Zeile 601 - 627

```
1  case PHSYSICS_RENDERMODE_HEUN:
2          for(itP = ListProtone.begin(); itP!= ListProtone.end(); itP++ ) {
3                  (*itP)->Speed += (*itP)->SpeedingUp * 0.0002f * 0.5f;
4                  (*itP)->Position += (*itP)->Speed * 0.5f;
5          }
6          for(itE = ListElectrone.begin(); itE!= ListElectrone.end(); itE++ ) {
7                  (*itE)->Speed += (*itE)->SpeedingUp * 0.200f * 0.5f;
8                  (*itE)->Position += (*itE)->Speed * 0.5f;
9          }
10         ...
11         break;
12
13         // Berechung der aktuellen Beschleunigungen
14
15 case PHSYSICS_RENDERMODE_HEUN:
16         for(itP = ListProtone.begin(); itP!= ListProtone.end(); itP++ ) {
17                 (*itP)->Speed += (*itP)->SpeedingUp * 0.0002f * 0.5f;
18                 (*itP)->Position += (*itP)->Speed * 0.5f;
19         }
20         for(itE = ListElectrone.begin(); itE!= ListElectrone.end(); itE++ ) {
21                 (*itE)->Speed += (*itE)->SpeedingUp * 0.200f * 0.5f;
22                 (*itE)->Position += (*itE)->Speed * 0.5f;
23         }
24         ...
25         break;
26         }
27 }
```

6 Versuchsaufbauten

6.1 Wasserstoffatom

Der einfachste Versuchsaufbau ist wohl ein Elektron, das um das 1000-fach schwerere Proton kreist. Natürlich ist dies nur eine primitive Vereinfachung des Wasserstoffatoms, welche alle bohrsche Quantenbedingungen und sonstige Atomvorstellung außen vor lässt. Zu sehen ist allerdings die Ellipsenbahn, auf der sich das Elekton bewegt:

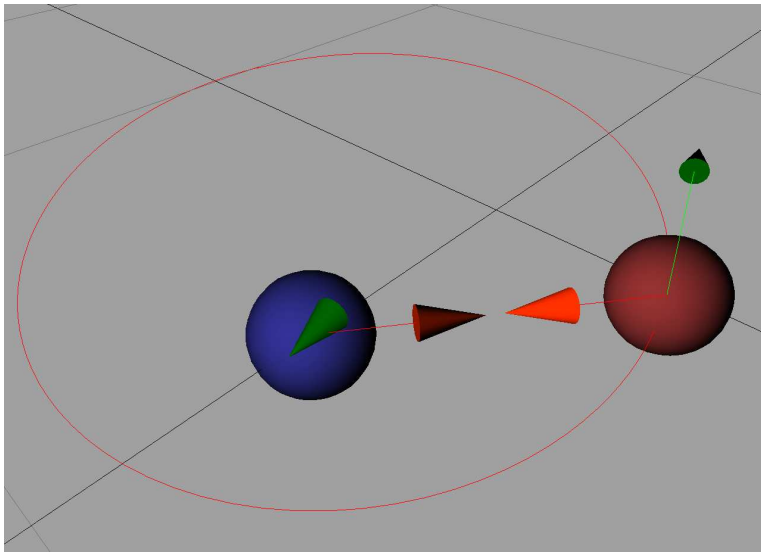

Abbildung 7: Orbit-Versuch

Man beachte, dass jeweils die Impulse und die Kräfte entgegengesetzt sind, und vom Betrag her gleich groß sind.
(Dieser Versuch ist auf der CD unter dem Verzeichnis *Beispielszenen/H-Atom.sim* zu finden)

6.2 Rutherford-Versuch

Im Jahre 1911 beschoss Ernest Rutherford eine dünne Goldfolie mit Alphateilchen. Aus diesen Ergebnissen begründete er ein neues nach ihm benanntes Atommodell. Wenn man das Experiment genauer unter die Lupe nimmt, erkennt man, dass Rutherford positiv geladene Teilchen auf einen Atomkern schleuderte, welche von der *Coloumb-Kraft* abgelenkt wurden. Ersetzt man die Alphateilchen durch Elektronen und gibt dem Atomkern eine negative Ladung, lässt sich auch die Rutherfordstreuung mit der Simulation zeigen.
(Zu finden auf der CD unter *Beispielszenen/RutherfordEx.sim*)

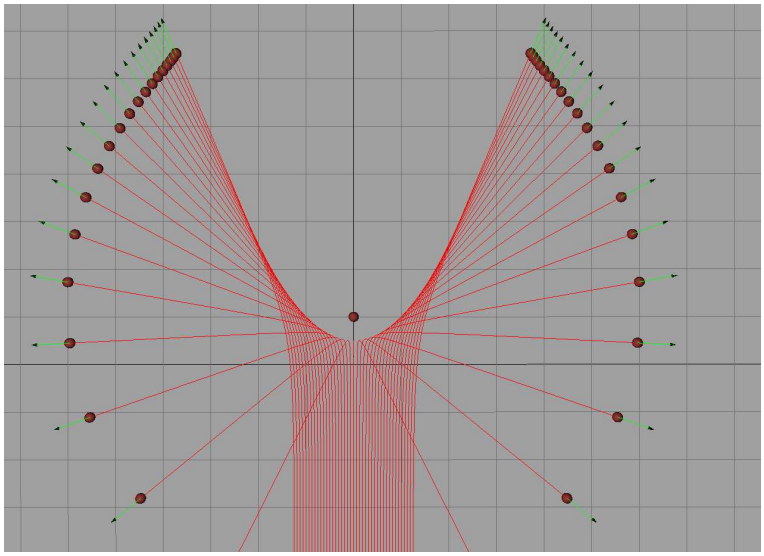

Abbildung 8: Rutherfordstreuung

6.3 WIEN-Filter

Ein ebenso interessanter Versuch ist der *Wien-Filter*. Dieses Gerät wird in der experimentalen Physik dazu verwendet, Teilchen mit der falschen Geschwindigkeit aus einem Strahl heraus zu sieben.

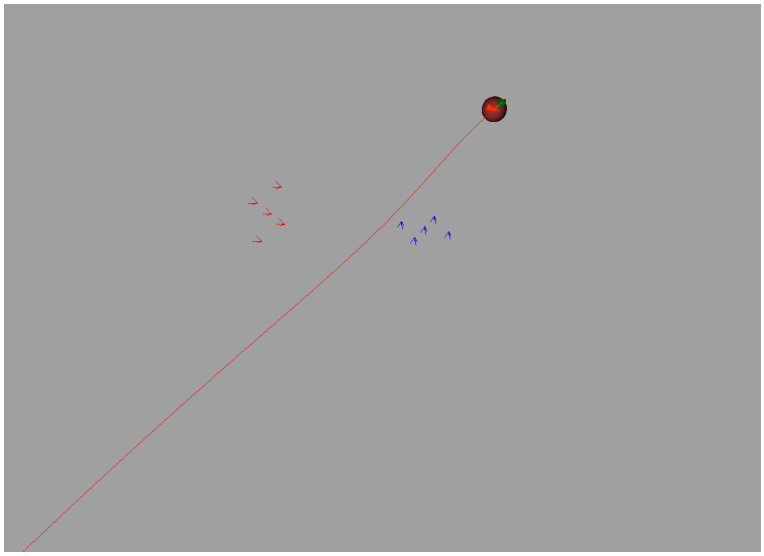

Abbildung 9: WIEN-Filter

Man beachte, dass die Bahn des Elektrons nicht exakt geradlinig ist, was daran liegt, dass die Anfangsgeschwindigkeit nicht 100-prozentig stimmt. Deshalb pendelt es immer zwischen der Übermacht der Lorenzkraft und der elektrischen Kraft.
(Zu finden auf der CD unter *Beispielszenen/wien-filter.sim*)

7 Grenzen computergestützter Simulationen

Doch leider wird man bei der Entwicklung computergestützter Simulationen immer wieder mit der Tatsache konfrontiert, wie schnell die begrenzte Rechenleistung eines Heim-PCs und die schwierige Beschreibung der Natur in manchen Teilbereichen jemanden an die Grenzen der physikalischen Simualtionstechnik treibt. Simulationen beispielsweise von Interferenzen an einem Doppelspalt würden bei einer üblich hohen Auflösung in Elementarwellen, die CPU-Leistung um ein Vielfaches überanspruchen. Vor allem wenn es darum geht, ganze Volumina mit Werten zu füllen, wie es z.B. bei dem Magnetfeld eines Helmholzspulenpaares der Fall wäre. Aus diesem und anderen Gründen wurde auf dieses Gerät im Programm verzichtet. Allein das Berücksichtigen des Luftwiderstandes bei mechanischen Versuchen würde jede Computersimualtion den Gar ausmachen. Ebenfalls ist man nie davor gewappnet, irgendwelche Umwelteinflüsse zu vergessen. Was die Frage der Verlässlichkeit nicht zu gunsten der Simulationen beantwortet. Betrachtet man die heißelbergsche Unschärferelation, dann wären die meisten Versuche mit Mirkoteilchen nicht mehr ganz so trivial zu beschreiben ,wie es der Computer mit seinen Schaltkreisen vermag. Daher ist nicht jedes Gebiet der Physik gleich gut geeignet, es in ein PC-Programm zu verpacken.

Abbildungsverzeichnis

Listings

Verwendete Quellen

1. http://www.stmwfk.bayern.de/pressearchiv/meldung.asp?NewsID=649 3

2. http://www.medien.ifi.lmu.de/fileadmin/mimuc/mll ws0506/Vortrag Kim.pdf 4

3. Physik, Leistugskurs 1. Semester, Seite 61 8

4. Physik, Leistugskurs 1. Semester, Seite 110/111 9

5. http://www-history.mcs.st-andrews.ac.uk/Biographies/Heun.html 14

6. http://www.acdca.ac.at/material/kl8/numerik.pdf 15

Ich erkläre hiermit, dass ich die Facharbeit ohne fremde Hilfe angefertigt und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel benutzt habe.

_____, den	_____	_____
Ort	Datum	Unterschrift