Simpert-Kraemer-Gymnasium **2007/2008**
Krumbach

Facharbeit

aus dem Fach

Physik

Thema: Computersimulation von geladenen Teilchen im elektrischen und magnetischen Feld

Verfasser:	Lochbrunner Matthias
Leistungskurs:	Physik
Kursleiter:	Herr Urban
Abgabetermin:	23. Dezember 2007

Erzielte Note: _____ In Worten: _____

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

Abgabe im Sekretariat: _____

Unterschrift des Kursleiters: _____

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

1 Einführung zum Thema

Spätersten nach der Begründung der Quantenphysik durch Albert Einstein sind computergestützte Simulationen in den Naturwissenschaften kaum mehr wegzudenken. Diese Hilfe verwendet man "'in der Elementarteilchenphysik, in der Materialforschung, Strömungsdynamik, Strukturmechanik, Chemie, Geo- und Astrophysik sowie Klima- und Umweltforschung"'[1]. Dazu werden heutzutage 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. Nicht selten treffen Lehrer in Sachen wie Elektrizität bei Schülern auf Unverständis und mangelnder Vorstellungskraft. 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. Diese Nische zu füllen entschied ich mich, selber im Rahmen dieser Facharbeit, ein Programm zu entwickeln, das auf der Basis von Microsoft DirectX^{TM}geschrieben in *C++*, physikalische Experimente anschaulich im Unterricht vorführen kann.

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 gegebenfalls, die "'Runtime"' Version auf dem PC zu aktualisieren [2].
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 erhhält das Programm nun im Startmenü, unter *alle Progamme* einen Eintrag.

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

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

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 Urprung 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 links unten hineingezoomt. Allerdings ist zu beachten, dass der Zoom nicht linear zur Mausbewegung berechnet wird, sondern exponential. So wird ein für die meisten Situationen praktisches und gleichsam harmonisches Zoomen ermöglicht. Des Öftern kann es passieren, dass man sich irgendwo im Raum verliert und keine Orientierung mehr hat. Hierfür empfiehlt sich der Hotkey F , welcher die Kamera auf das zuletzt markierte Element zentriert bzw. auf den Ursprung.

2.3 Arbeiten im dreidimensionalen Raum

In der Simulation stehen zwei Werkzeuge (englisch: "'Tools"') zu 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 zu Unregelmäßigkeiten kommen kann.

2.4 Arbeiten mit der "'Channelbox"'

Um genaue Angaben zu einem Objekt machen zu können, wurde ein Kontrollfenster, oder im Fachchargon auch "'Channelbox"' genannt, erstellt, mit dem der Benutzer in der Lage ist jeden Wert genau ...

2.5 Szene animieren

F2 starten / stoppen TAB schrittweiße ...

2.6 Spur anzeigen

...

3 Eingebettete Phyikalische Gesetze

3.1 Elektrisches Feld

3.1.1 Physikalischer Hintergrund

...

3.1.2 Programmiertechnische Umsetzung

...

Listing 1: Quellcodeausschnitt aus Physics.cpp, Zeile 756 - 765

```
1  D3DXVECTOR3 CPhysics::ForceElectrical(D3DXVECTOR3 DrivenPosition, D3DXVECTOR3 DriverPosition){
2          D3DXVECTOR3 v;
3          float DistanceInSquare = (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) / DistanceInSquare / sqrt(DistanceInSquare);
8
9          return v;
10 }
```

3.2 Magnetisches Feld

3.2.1 Physikalischer Hintergrund

...

3.2.2 Programmiertechnische Umsetzung

...

Problem der endlichen Zeiteinteilung...

Listing 2: Quellcodeausschnitt aus Physics.cpp, Zeile 768 - 783

```
1  D3DXVECTOR3 CPhysics::ForceMagnetical(D3DXVECTOR3 DrivenSpeed, D3DXVECTOR3 DriverForce){
2          D3DXVECTOR3 v = D3DXVECTOR3(0.0f, 0.0f, 0.0f);
3
4          D3DXVECTOR3 temp = D3DXVECTOR3(0.0f, 0.0f, 0.0f);
5
6          v.x += +DrivenSpeed.z * DriverForce.y + DrivenSpeed.x * DriverForce.y * DriverForce.y *
               MAGNETICAL_HOLD;
7          v.z += -DrivenSpeed.x * DriverForce.y + DrivenSpeed.z * DriverForce.y * DriverForce.y *
               MAGNETICAL_HOLD;
8
9          v.y += -DrivenSpeed.z * DriverForce.x + DrivenSpeed.y * DriverForce.x * DriverForce.x *
               MAGNETICAL_HOLD;
10         v.z += +DrivenSpeed.y * DriverForce.x + DrivenSpeed.z * DriverForce.x * DriverForce.x *
               MAGNETICAL_HOLD;
11
12         v.x = -DrivenSpeed.y * DriverForce.z + DrivenSpeed.x * DriverForce.z * DriverForce.z *
               MAGNETICAL_HOLD;
13         v.y = +DrivenSpeed.x * DriverForce.z + DrivenSpeed.y * DriverForce.z * DriverForce.z *
               MAGNETICAL_HOLD;
14
15         return v;
16 }
```

4 Objekte

4.1 Elementarteilchen

...

4.2 Homogene Felder

...

4.3 Metallkugel

...

4.4 Kondensatorplatte

...

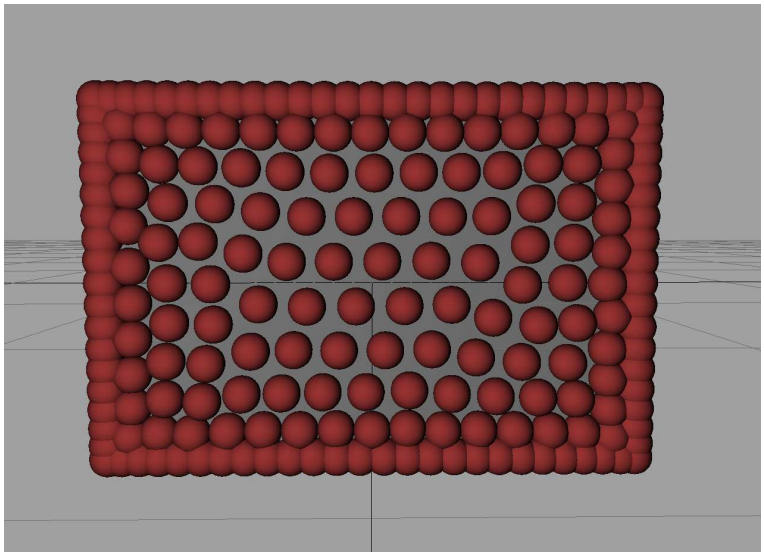

Abbildung 1: Kondensatorplatte

4.5 "'Emitter"'

...

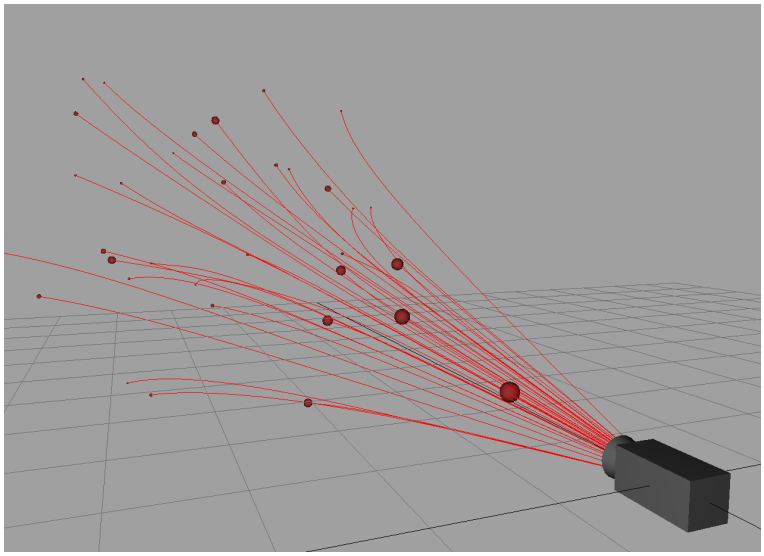

Abbildung 2: Emitter

5 Halbschrittverfahren

5.1 Verfahren von Heun

...

$$\begin{array}{lcll} \text{Es sei gegeben die Funktion} & : & f(x) & = & m \cdot x + t \\ \text{Auf Grund der Linearität} & : & f(x_1 + \Delta x) & = & f(x_1) + f'(x_1) \cdot \Delta x \\ \text{Daraus folgt nach Heun}^3 & : & f(x_1 + \Delta x) & = & f(x_1) + \frac{f'(x_1)+f'(x_1+\Delta x)}{2} \cdot \Delta x \end{array}$$

...
RUNGE-KUTTA-Verfahren

5.2 Programiertechnische Anwendung

...

6 Versuchsaufbauten

...

6.1 Culomb-Kräfte

...

6.2 Rutherford-Versuch

...

6.3 WIEN-Filter

...

7 Grenzen computergestützer Simulationen

...

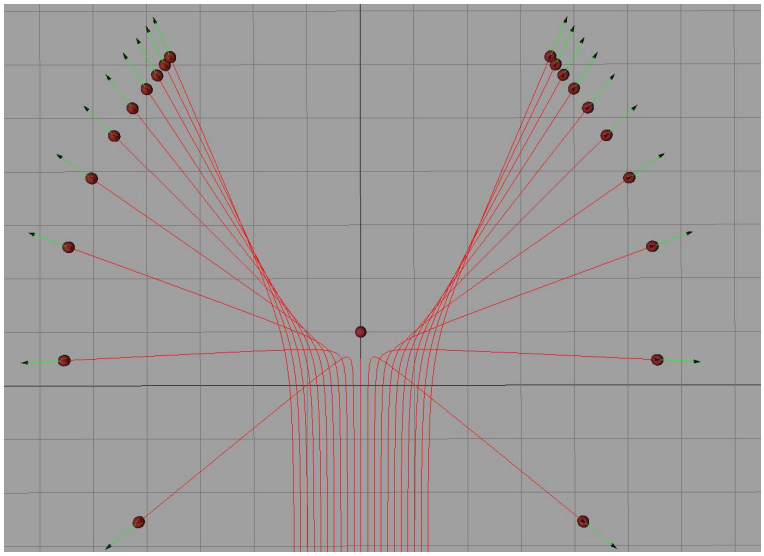

Abbildung 3: Rutherfordstreuung

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