Kurs 5 – Billard: Auf der Suche nach dem perfekten Stoß

Unser Kurs

Vorwort

Die Kursleiter

„Endlich Sommerferien! Ab in den Urlaub!“, denken sich so manche Schüler, die in den so lang ersehnten Sommer entlassen werden. Die Science Academy lässt sich natürlich nicht mit Sandstrand und Sonnenbaden vergleichen: Das Erlebnis, da wird man uns zustimmen, ist jedoch um einiges wertvoller. In unserem Kurs, dem Physik-Informatik-Kurs, dem „Billardkurs“, dem „coolsten Kurs“, beschritten zwölf Teilnehmer aus ganz Baden-Württemberg gemeinsam einen Weg, um das Ziel, ein Billardspiel physikalisch genau zu untersuchen und es nach diesen Regeln auf dem Computer realistisch zu programmieren, zu erreichen. Dazu mussten gewisse Grundkenntnisse, aber auch Spezialwissen vermittelt und selbstständig angewendet werden. Durch diese intensive Zusammenarbeit entstanden Vertrauen und Freundschaften – nicht ohne Grund hat unser Kurs das Sportfest gewonnen. Für den Spaß hatten wir einen üppigen Süßigkeitenvorrat, einen Billardtisch, Experimente und gelegentlich lustige Bugs in unserem Programm. Auch wir Leiter werden diese zwei Wochen immer in guter Erinnerung behalten.

Unser Kurs

Myriam Holl, Verena Winterhalter

David: Unser „Superbrain“ hat sich vom Eröffnungswochenende bis zur Akademie selbst übertroffen und ließ uns alle staunen, wenn er mal wieder zeigte, wie schnell er Gelerntes umsetzen konnte. Seine Programme waren zwar von einer nicht für alle offensichtlichen Ordnung durchzogen, doch solange er den Überblick behielt, sollte uns das egal sein. In seiner Freizeit traf man David häufig beim Programmieren mit Zettel und Stift an.

Frank: Bekannt als „Franky-Boy“, fand er es lustig, Namensschilder zu tauschen und dadurch den einen oder anderen zu verärgern. Er verbreitete immer gute Laune im Kurs und hatte die Angewohnheit, jeden zum Lachen zu bringen und mit sich selbst zufrieden zu sein. Er hat übermäßig viel daran gesetzt, unseren Billardtisch mit Diamanten zu verschönern, was uns eine bis dahin unbekannte Seite von Franks vielseitigem Charakter gezeigt hat. Sein Sieg zusammen mit Lukas beim kursinternen Billardturnier gehörte zu einem der Akademiehöhepunkte.

Laura: Unsere einzigartige Meisterin im Stellen gemeiner Fragen hat Alex an den Rand der Verzweiflung getrieben, wenn sie etwas ganz genau wissen wollte. Da musste Alex wohl oder übel eine Nacht lang grübeln, bevor er Laura die Antwort präsentieren konnte. Klasse Leistung, Laura! Im Allgemeinen war Laura eine Bereicherung für jede Gruppenarbeit, da sie sich mit viel Freude, Können und Hilfsbereitschaft an jedes noch so große Problem gewagt hat.

Ludwig: Durch das Forum als „König Ludwig“ bekannt, offenbarte er während der Akademiezeit einige unerwartete Aspekte seiner selbst, darunter das Ohrenwackeln oder die gekonnte Moderation durch den Hausmusik-Abend. Als Spaßvogel war er ein weiterer Kandidat, der an den Nerven unserer Kursleiter zog und zerrte. Wir freuten uns aber immer über Ludwigs auflockernde Kommentare, die man von ihm auf Grund seines ruhigen Wesens so nicht erwartet hätte.

Lukas: „Mr Photoshop“ ist ein wahres Physik-Ass und verstand jede Herleitung, die Alex uns auftischte, sofort und stellte Fragen dazu, bevor andere überhaupt die Sachlage überblickt hatten. Auch außerhalb der Kurszeit arbeitete er fleißig an unserem Spiel. Sein Sieg zusammen mit Frank beim Billardturnier hat ihm auch lokale Publicity eingebracht, sowie eine Prinzenrolle.

Unser Kurs-T-Shirt und unser Kurspullover

Myriam: „Mimi“ war immer gut drauf und hatte ein Talent zum Präsentieren. Egal ob Programme funktionierten oder nicht, Mimi achtete stets bei sich und anderen darauf, dass alles ordentlich und übersichtlich war. Ihre gute Laune war fast schon gefährlich ansteckend und ihr Modebewusstsein hat im Laufe der Akademie auf andere Kursteilnehmer abgefärbt. Durch ein dringendes Bedürfnis vor der Abschlusspräsentation wurde Adelsheim Down-Town durch Mimi für „ungenügend ausgerüstet“ befunden, denn nirgends waren Lockenwickler aufzufinden. Ihre legendäre Lockenpracht saß dann am entscheidenden Termin aber wie immer perfekt.

Sebastian: „Sebbi“ hat beim Moderieren des Bergfests mit seinem Charme alle in seinen Bann gezogen, ebenso beim Halten der Abschlusspräsentation. Sportbegeistert wie er ist, war Sebbi bei jeder Tanz-KüA mit von der Partie und beim Joggen musste man nie auf ihn warten. Trotz morgendlichen Frühaufstehens war er immer gut gelaunt und für jeden Spaß zu haben. Die Tatsachen,

dass er gut erklären kann und jederzeit hilfsbereit ist, machten ihn zu einem wertvollen Kursmitglied. Seinem einmaligen Lächeln konnte sich keiner entziehen, sodass er ein Garant für gute Laune war.

Unser Kursmaskottchen Sigmund

Sigmund: Das wichtigste Kursmitglied hat uns bei den Vorträgen moralisch unterstützt wie kein anderer. Ohne ihn wäre unser Ausflug ins Interdisziplinäre Zentrum für Wissenschaftliches Rechnen bestimmt nur halb so spannend gewesen und bei der Wanderung half er uns zuverlässig, den richtigen Weg zu finden. Seinen Aufgaben als Kursmaskottchen mit eigenem Facebook-Account ist Sigmund zu vollster Zufriedenheit nachgekommen und wir werden ihn alle sehr vermissen.

Tatjana Als Ruhepol unseres Kurses hatte sie gute Ideen und ihre einzigartige Fähigkeit, über alles und jeden bis zum Umfallen diskutieren zu können, hat beim Mittagessen durchaus zu langen Gesprächen geführt. Ihre unkomplizierte und hilfsbereite Art machte sie zu einem Menschen, mit dem man gerne mehr Zeit verbrachte. Tatjana ist eine interessante Person und war eine große Bereicherung für unseren Kurs.

Theresa oder „Verena“: Sie war wie besessen auf der Suche nach passenden Texturen für unser Spiel. Und natürlich wurde sie hier fündig, ebenso bei der Suche nach Lösungen zu Alex Kopfnüssen des Tages. Kein einziges Rätsel war vor Theresas hellem Köpfchen sicher. Sie hat im Kurs immer alles schnell verstanden und konnte ihr Wissen auch gut mit anderen teilen. Außerdem hat Theresa das Design und die Bestellung der Kurs-T-Shirts wunderbar organisiert.

Tim: Er ist sehr musikalisch und wurde deshalb auch „The Soundmachine“ genannt. Sein musikalisches Können stellte er mehrmals zusammen mit der Akademie-Band „Diversion“ zur Schau. Er hatte immer gute Ideen, was wir in unser Programm einbauen könnten und setzte diese selbstständig um. Dank seiner PowerPoint-Kenntnisse erstellte er alle Grafiken für unsere Dokumentation.

Verena oder „Theresa“: Sie war unsere Designerin schlechthin. Egal ob es um ihre Fingernägel oder um die Schatten unserer Kugeln ging, sie erledigte es im Handumdrehen und war hauptverantwortlich für das tolle Aussehen unseres Programms. Mit ihr hat man gerne seine Freizeit verbracht, und man konnte mit ihr gut diskutieren. Bei der Abschlusspräsentation wollten wir sie als unseren „Kurszwerg“ auf ein Podest stellen, aber durch ihre Absatzschuhe und ihr selbstbewusstes Auftreten haben wir davon dann doch abgesehen.

Wendy: Sie war unser kleiner Sonnenschein, der sowohl in der Zeitungs-KüA als auch im Kurs immer für gute Laune sorgte. Sie schaffte es, unser Spiel perfekt zu präsentieren, vor allem ihre Einführungen waren legendär. Sie hat einen ausgeprägten Sinn für Sarkasmus, der uns immer wieder zum Lachen brachte. Wendy hatte immer viel Spaß und Teamfähigkeit beim Arbeiten in Gruppen gezeigt und konnte sehr gut auf Ideen und Vorschläge anderer eingehen.

Unsere Kursleiter

Alex: Der coolste Kursleiter war als begeisterter Snooker-Schauer bei uns im Kurs goldrichtig. Leider sind noch bessere Billardspieler als er im Kurs gelandet, also musste er sich mit Daniel gegen Frank und Lukas im Billardturnier geschlagen geben. Als Rhetorikbeauftragter hat er uns und der ganzen Akademie einiges über Präsentationen und doofe Biber erzählt. So spannend wie Alex kann keiner moderne Physik erklären, und auch mit den Kopfnüssen des Tages hat er uns auf Trab gehalten.

Daniel: Als ebenfalls coolster Kursleiter und „alter Hase“ der Akademie hat er uns sehr gut alles Erdenkliche an Programmierkenntnissen beigebracht. Er war sehr überrascht, was wir in zwei Wochen alles in unser Spiel eingebaut hatten. Daniels Tanzkünste haben sich auch bis zu uns in den Kurs durchgesprochen, er soll ein hervorragender Standard und Latein-Tänzer sein. In der kursfreien Zeit traf man Daniel beim Marathontraining an oder man erwischte ihn dabei, wie er schamlos fragte, wie breit so ein Spielfeld eigentlich sei.

Irina: Liebevoll „Mama“ genannt, sorgte sie immer für ein sehr gutes Kursklima. Ihr entging kein vergessenes Semikolon; und beim Sportfest hat sie einen großen Teil dazu beigetragen, dass der Physik-Kurs auch dieses Jahr erfolgreich den Titel geholt hat. Sie war immer für uns da, hatte ein offenes Ohr für alle und wir haben viel mit ihr gelacht, vor allem wenn sie in Erinnerungen an ihre eigene Akademiezeit schwelgte. Besonders die Tatsache, dass Irina immer an unsere zuckersüße Versorgung gedacht hat, machte sie unverzichtbar für die zweiwöchige Akademie.

Unser Spiel

Wendy Yi

Wer einige Tage vor dem Ende der Sommerakademie in den Raum unseres Kurses kam, hörte oft ein für Außenstehende unverständliches Jubeln und Kommentare wie: „Juhu, endlich eingelocht!“ oder auch „Gewonnen!“. Uns hatte nämlich eine merkwürdige Sucht befallen: Das Billardspielen am Computer. Wir versuchten alle verzweifelt, die Kugeln in die Löcher zu stoßen, was nicht immer so einfach war. Denn nach zwei anstrengenden, aber auch lustigen und lehrreichen Wochen voller Physik, Mathematik und Informatik, hatten wir endlich unser Ziel erreicht: Ein physikalisch realistisches und objektorientiert programmiertes Billardspiel, das sich sehen lassen kann.

Die endgültige Version unseres Billardspiels besteht nun aus insgesamt 16 Kugeln (mit 3D-Effekt!) – 15 farbigen und einer weißen. Da man acht Kugeln lochen muss, um zu gewinnen, nennt man diese Art von Billard „8-Ball-Spiel“. Dabei gibt es sieben volle, also komplett farbige Kugeln und ebenfalls sieben halbe Kugeln sowie die schwarze Acht. Alle Kugeln haben jeweils eine andere Farbe, eine andere Zahl und haben am Anfang des Spiels eine feste Position, genauso wie bei einem richtigen Billardspiel:

Natürlich können die Kugeln sich auch bewegen. Damit haben wir uns eine ganze Weile beschäftigt: Sie bewegen sich nicht nur von einem Punkt zum anderen, nein, sie rollen auch noch – und das sehr realistisch. Damit die Kugeln

aber nicht ewig weiterrollen und unser schöner Reibungsversuch nicht umsonst war, werden die Kugeln wirklichkeitsgetreu langsamer.

Zu Beginn hatten wir gelegentlich mit der Kollision Probleme, da die Kugeln manchmal durch einander hindurch rollten. Am Ende aber funktionierte alles so wie bei einem echten Billardspiel: Das Programm erkennt jede Kollision zuverlässig und die Kugeln prallen von der Bande ab, und zwar nach dem Gesetz „Einfallswinkel gleich Ausfallswinkel“, und auch die Kollisionen der Kugeln untereinander sehen aus wie auf einem echten Billardtisch.

Bei einem Billardspiel geht es ja hauptsächlich um das Einlochen der Kugeln und das macht bei unserem Spiel besonders Spaß: Denn die Kugeln verschwinden nicht einfach langweilig in den Löchern, sondern werden, wenn man genau hinschaut, erst langsam kleiner, bis sie endgültig verschwinden. Das erweckt den Eindruck, als würden sie wirklich hinein fallen. Und als Sahnehäubchen oben drauf werden die eingelochten Kugeln oben am Rand angezeigt, damit man nie vergisst, wer gerade am Gewinnen ist. Natürlich soll die weiße Kugel nicht verschwinden, wenn sie eingelocht wird, da man ansonsten nicht mehr weiterspielen kann. Deshalb kann man diese nach dem Einlochen wieder auf den Tisch setzen – wie bei einem richtigen Billardspiel. Dabei kann man die Kugel auf einer gedachten Linie mit der Maus nach oben oder unten verschieben, wobei es nicht möglich ist, sie auf eine andere Kugel zu setzen.

So weit, so gut. Allerdings brauchen wir auch noch etwas zum Anstoßen, da die Kugeln ja nicht von alleine anfangen zu rollen. Die Lösung: der Queue! Auf unseren Queue sind wir alle besonders stolz. Bei den Präsentationen hatten wir den echten Queue immer zum Vergleich daneben gehalten, denn es herrscht eine verblüffende Ähnlichkeit – obwohl unser Queue natürlich noch viel besser aussieht als der echte.

Der Queue erscheint nur, wenn alle Kugeln ruhen und mit ihm kann man natürlich nur die weiße Kugel anstoßen. Die Steuerung des Queues ist ziemlich einfach: Man kann den Queue mit der Maus steuern, indem man die Richtung mit der Maus vorgibt. Je weiter die

weiße Kugel von dem Mauszeiger entfernt ist, desto stärker stößt der Queue.

Auch der Billardtisch ist einem echten nachempfunden. Der Hintergrund stammt zum Beispiel von einem abfotografierten Billardtuch und auch die Maße des Tisches und die Positionen der Löcher sind wie bei einem echten Tisch. Manch einer hält die Diamanten am Rand des Tisches vielleicht nur für ein oberflächliches Detail, allerdings hat es einige Jungs unseres Kurses eine ganze Weile gekostet, bis sie die „perfekten“ Diamanten – Theresas Ohrringe – gefunden, abfotografiert und entsprechend bearbeitet hatten. Und wenn man einfach nicht genug von dem Spiel kriegen kann, weil es einfach das coolste Billardspiel vom absolut coolsten Kurs ist, und man es immer und immer wieder spielen möchte, so drückt man einfach die Taste „n“ für „Neustart“ – immer und immer wieder.

Man kann sich bestimmt denken, dass hinter unserem Spiel ein langer Weg voller Programmierarbeit, physikalischer Formeln und natürlich auch Spaß steckt. Diesen Weg haben wir auf den folgenden Seiten dokumentiert.

Der Weg zu unserem ersten Programm

Laura Merker

Eröffnungswochenende

Um uns gut auf die zwei Wochen im Sommer vorzubereiten, trafen wir uns zehn Wochen vor der Akademie schon einmal zum Eröffnungswochenende. Hier lernten wir nicht nur die anderen Teilnehmer und Kursleiter kennen, sondern

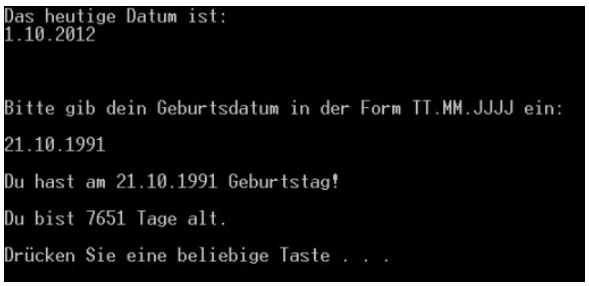
auch die Programmiersprache C++, mit der wir bald ebenso gut zurechtkamen wie mit den anderen Teilnehmern und unseren Kursleitern.

Da einige von uns noch nie zuvor programmiert hatten, stellten wir uns zuerst die Frage: Was ist Programmieren überhaupt? Natürlich wollen wir dem Computer Befehle geben, aber er muss sie ja auch verstehen. Das kann er nicht sofort, sondern er muss sich den Quelltext, in dem unsere Befehle stehen, zuerst vom Compiler übersetzen lassen. Der Compiler – ein spezielles Programm – ist also dafür da, unser Programm in Maschinensprache, die nur aus Nullen und Einsen besteht, zu übersetzen. Als wir das soweit geklärt hatten, wollten wir natürlich sofort loslegen. Doch bevor wir uns an das Billardspiel wagen konnten, mussten wir erst einmal ein paar Grundkenntnisse erlernen.

Eines unserer ersten Programme ließen wir beispielsweise so lange mit einem herkömmlichen Würfel würfeln, bis es eine Sechs würfelte. Das hört sich zwar einfach an, aber wir lernten viel Neues, das wir später für unser Billardspiel gebrauchen konnten. Zum Beispiel lernten wir Variablen kennen, mit denen wir Zahlen, Buchstaben oder andere Werte speichern konnten. Wichtig war aber auch die Ein- und Ausgabe, d. h. der Computer konnte uns Ergebnisse einer Rechnung mitteilen oder wir konnten ihm Daten übergeben. Außerdem lernten wir, wie wir Befehle beliebig oft mit Hilfe einer Schleife ausführen oder eine Bedingung stellen können. Zum Ende des Eröffnungswochenendes waren wir dann schon so weit, dass wir einen Billardtisch und Kugeln, die wir vorerst mit einfachen Kreisen darstellten, zeichnen konnten. Dazu hatten wir uns gemeinsam überlegt, wie die Kugeln gegen die Bande stoßen, und vor allem, wie wir dies im Programm umsetzen, sodass sich die Kugeln auf dem Tisch bewegen konnten.

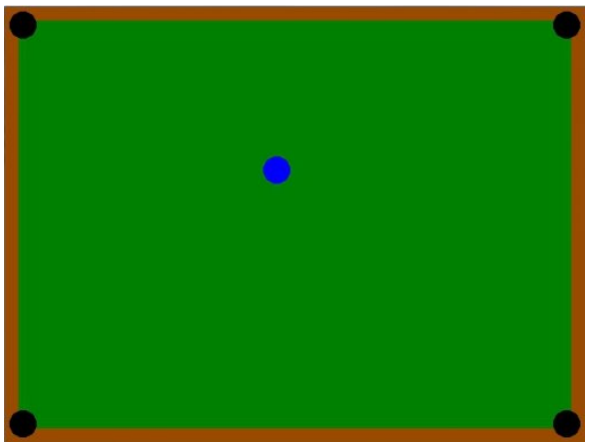
Vorkurs

Um das Erlernte bis zum Sommer zu vertiefen, gab es bis zu den Ferien drei Übungszettel. Unsere erste Aufgabe war es, ein Programm zu schreiben, welches das Alter in Tagen berechnet. Dazu soll der Benutzer sein Geburtsdatum eingeben, das aktuelle Datum kann das Programm selbst ermitteln.



Unser Programm berechnet Sigmunds Alter

Außerdem schrieben wir ein Programm, das uns sagt, ob eine Zahl eine Primzahl ist oder nicht und beschäftigten uns mit Texten, Farben und Formen, wie zum Beispiel einer Spirale, um sie in einem Fenster anzeigen zu lassen. Den Abschluss unserer Übungen bildete ein Billardtisch, bei dem die Kugeln in Löcher fallen konnten.



Eine sehr frühe Version unseres Billardspiels

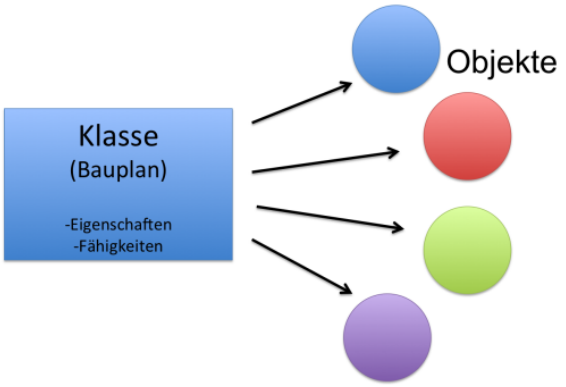
Natürlich ging es bei den Aufgaben nicht nur um Spiralen und Primzahlen, sondern darum, dass wir unsere Programmierkenntnisse erweiterten. So lernten wir zum Beispiel Funktionen kennen, mit deren Hilfe man den Quelltext übersichtlich strukturieren kann. Der Vorteil von Funktionen ist, dass wir sie nicht nur einmal, sondern beliebig oft aufrufen können, ohne alles nochmal abtippen zu müssen. Damit waren wir unserem Ziel, einem eigenen Billardspiel, schon einen großen Schritt näher gekommen und hatten den Grundstein für unsere erfolgreiche Arbeit im Sommer gelegt.

Objektorientierte Programmierung

Frank Wolff

Später in unserem Billardspiel wollen wir mehr als eine Kugel erzeugen. Diese sollen aber nicht im Hauptprogramm einzeln programmiert werden, da das sehr aufwändig und unübersichtlich wäre. Alle Kugeln besitzen Eigenschaften wie Koordinaten, Geschwindigkeit oder Farbe, aber auch die Kollisionen mit der Bande und die Bewegung sind bei allen Kugeln gleich. Wir greifen daher auf die Möglichkeiten, die uns die objektorientierte Programmierung bietet, zurück. Im Grunde hilft sie uns, genau diese gleichen Eigenschaften zusammenzufassen und sie aus unserem Hauptprogramm auszulagern, wodurch dieses viel besser lesbar und der Programmcode insgesamt kürzer wird. So können wir später mit einer einzigen Codezeile im Hauptprogramm ein ganzes Paket an Funktionen aufrufen.

Doch wie genau funktioniert das objektorientierte Programmieren? Nachdem man sich klar gemacht hat, was genau unser Objekt zum Auslagern ist und welche Variablen (Eigenschaften des Objekts) und Funktionen (Fähigkeiten des Objekts) es enthalten soll, werden diese Dinge in einer Art Bauplan als separate, sogenannte Header-Datei festgehalten und alle Variablen und Funktionen erst einmal definiert.



Neben dem Anlegen des Bauplans müssen die Funktionen natürlich programmiert werden, was in einer weiteren, separaten Programmdatei geschieht. Das eigentliche Bauen, also das Erzeugen einzelner Objekte, geschieht dann im Hauptprogramm. Dort werden auch alle nötigen Variablen der Objekte initialisiert und ihre Funktionen aufgerufen. Schauen wir uns das am Beispiel einer Kugel aus einer vereinfachten Billard-Version an. Zunächst unser Bauplan:

```
class Kugel {

// Eigenschaften einer Kugel:
private:

  float radius;

  // Koordinaten:
  float x;
  float y;

  // Geschwindigkeit:
  float vx;
  float vy;

// Funktionen einer Kugel:
public:

  void bewegen(double t);
  void kollision(Spielfeld& s);
  void zeichnen();

};
```

Die Variablen sollen nur von der Kugel selbst geändert werden können und sind daher als privat (`private`) deklariert. Danach folgen die Funktionen, bei denen auch Objekte aus anderen Klassen, wie das Spielfeld, verwendet werden. Sie sind als öffentlich (`public`) deklariert, damit sie vom Hauptprogramm aufgerufen werden können. Exemplarisch sind hier zwei Funktionen aufgeführt, die – wie oben beschrieben – in einer separaten Datei implementiert werden:

```
void Kugel::bewegen(double t) {

// Aenderung der x-Koordinate
// der Kugel um vx * t:
  x += vx * t;

// Aenderung der y-Koordinate
// der Kugel um vy * t:
  y += vy * t;
}
```

```
void Kugel::zeichnen() {

// Farbe auf weiss setzen:
  glColor3f(1.0, 1.0, 1.0);

// Kugel zeichnen:
  graphicsBall(x, y, radius);

}
```

Jetzt müssen diese Funktionen nur an der richtigen Stelle des Hauptprogramms aufgerufen werden, exemplarisch zu sehen an der Zeichenfunktion:

```
void display() {

// 15 Kugeln zeichnen:
  for(int i = 0; i <= 14; i++)
    kugel[i].zeichnen();
}
```

Hier kann man ganz deutlich sehen, wie einfach wir nun im Hauptprogramm mit wenigen Zeilen Programmcode 15 Kugeln zeichnen können! So haben wir auch gleich noch den Nutzen unserer erfolgreich durchgeführten objektorientierten Programmierung anschaulich dargestellt und den ganzen Spuk um dieses Thema aufgelöst.

Billard und der Satz des Pythagoras

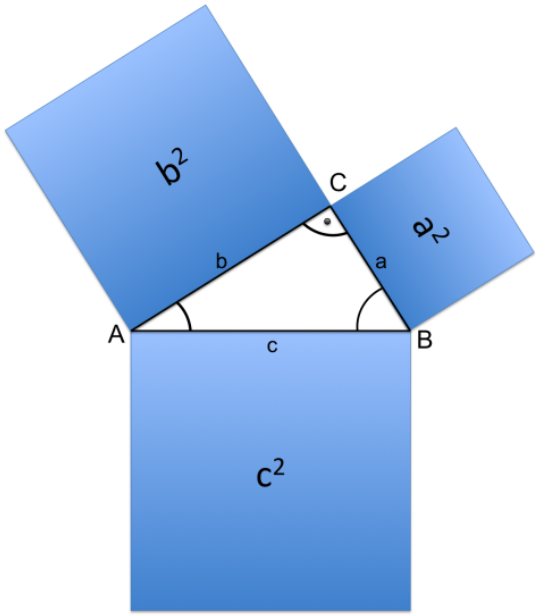
Theresa Häberle

Vom „Satz des Pythagoras“ hat fast jeder schon mal gehört, und auch, dass er

$$a^2 + b^2 = c^2$$

lautet. Aber was bedeutet das eigentlich? Und wozu braucht ihn ein Kurs, der sich mit Billard beschäftigt?

Wenn man bei einem rechtwinkligen Dreieck an jede Seite ein entsprechendes Quadrat anlegt, so ist der Flächeninhalt des Quadrats an der langen Seite (Hypotenuse) gleich der Summe der Flächeninhalte der Quadrate an den kurzen Seiten (Katheten). Dadurch kann man, wenn zwei Seitenlängen bekannt sind, die dritte ausrechnen.

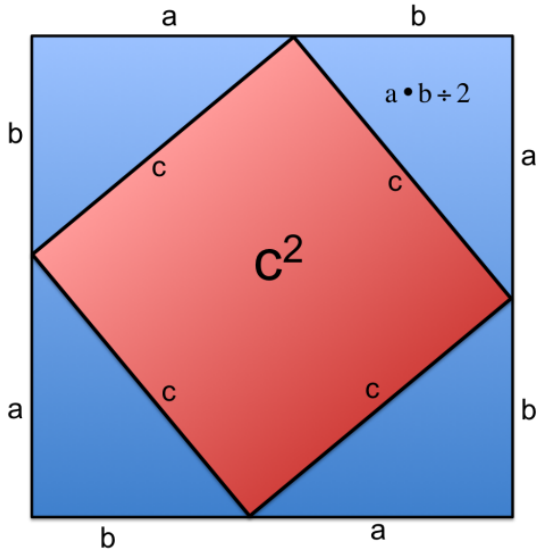


Der Satz des Pythagoras

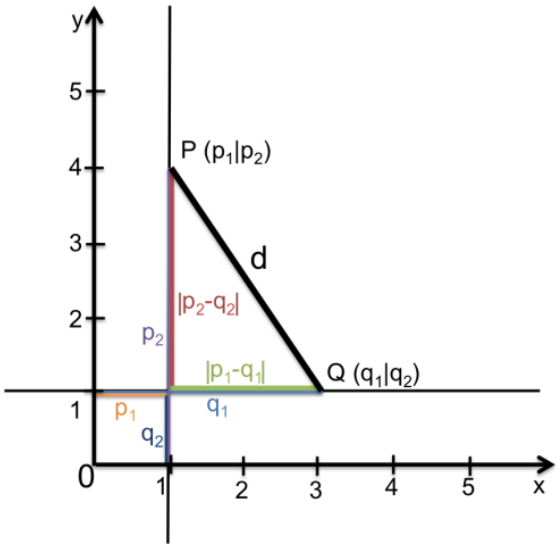
Beweis

An jeder Seite eines Quadrats wird die Strecke a abgetragen. So ergeben sich vier Punkte, die man zu einem Quadrat der Seitenlänge c verbinden lassen kann. Sein Flächeninhalt beträgt c^2. Außerdem entstehen vier rechtwinklige Dreiecke, deren Flächeninhalt jeweils $\frac{a \cdot b}{2}$ beträgt.

Der Flächeninhalt des großen Quadrats beträgt $(a+b)^2$ und ist gleich der Summe der Flächeninhalte des kleinen Quadrats und der vier Dreiecke:



Veranschaulichung des Beweises



Der Abstand zweier Punkte kann mit dem Satz des Pythagoras berechnet werden

$$(a+b)^2 = c^2 + 4 \cdot \frac{a \cdot b}{2}$$

Wir multiplizieren die Klammer aus:

$$a^2 + 2ab + b^2 = c^2 + 2ab$$

Jetzt ziehen wir auf beiden Seiten $2ab$ ab:

$$a^2 + b^2 = c^2 \qquad \square$$

Anwendung in unserem Programm: Abstand zwischen zwei Punkten

Da unser Billardtisch in einem Koordinatensystem liegt, wird die Position der Kugeln durch deren Mittelpunkte festgelegt. Wenn der Abstand zwischen zwei Mittelpunkten kleiner oder gleich der Summe beider Radien ist, erkennt der Computer die Kollision und löst die entsprechende Kollisionsbehandlung aus. Aber wie können wir den Abstand berechnen?

In einem Koordinatensystem liegen zwei Punkte P und Q, deren Koordinaten $(p_1 \,|\, p_2)$ und $(q_1 \,|\, q_2)$ bekannt sind. Wir wollen deren Abstand d errechnen:

Dazu zeichnen wir zwei Geraden ein, die parallel zur x- bzw. y-Achse sind und durch P bzw. Q verlaufen. Sie treffen rechtwinklig aufeinander. Der Abstand von der x-Achse zu den Punkten entspricht der jeweiligen x-Koordinate (p_1 bzw. q_1). Der Abstand von der y-Achse zu den Punkten entspricht der jeweiligen y-Koordinate (p_2 bzw. q_2). Es gilt:

- der Abstand von p_1 zu q_1 ist $|p_1 - q_1|$.
- der Abstand von p_2 zu q_2 ist $|p_2 - q_2|$.

Mit dem Satz des Pythagoras können wir jetzt den Abstand d berechnen. Es gilt

$$d^2 = (p_1 - q_1)^2 + (p_2 - q_2)^2$$

und damit

$$d = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}\,.$$

Hinweis: Es ist egal, ob man p_1 von q_1 oder andersherum abzieht, weil beim Quadrieren der Differenz das Vorzeichen ohnehin wegfällt.

Nach jeder Kugelbewegung wird für alle Kugeln der Abstand zueinander und zur Bande neu berechnet. Auf diese Weise können auftretende Kollisionen erkannt werden.

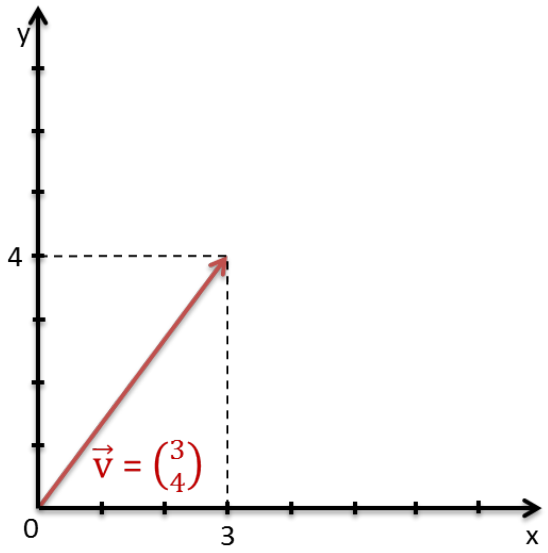
Vektorgeometrie

Tim Kopka

Wofür brauchen wir Vektoren?

In der Physik gibt es zwei unterschiedliche Arten von Größen, die sich fundamental unterscheiden. Viele Größen (wie z. B. Kraft oder Geschwindigkeit) haben eine Richtung, andere hingegen (z. B. Masse oder Volumen) besitzen keine Richtung. Größen mit Richtungen werden vektorielle Größen genannt und mit Hilfe von Vektoren angegeben, die durch mehrere Koordinaten beschrieben werden. Größen ohne Richtungen bezeichnen wir als Skalare – dargestellt durch einfache Zahlen. In unserem Spiel haben wir Vektoren für die Berechnung von Kugelkollisionen in zwei Dimensionen benötigt, damit diese ohnehin schon komplexe Rechnung zumindest einigermaßen überschaubar blieb.

Was sind Vektoren?



Definition eines Vektors

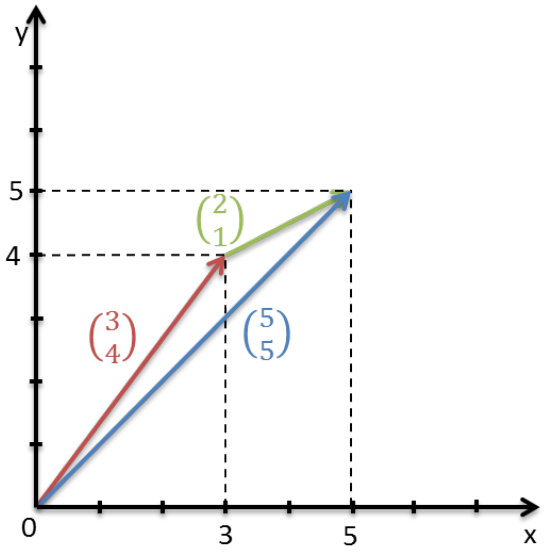
Wie bereits erwähnt, haben Vektoren eine Richtung, definiert durch mehrere Werte in unterschiedlichen Dimensionen $(x, y, z, ...)$ – im Gegensatz zu Skalaren, die nur aus einem Wert bestehen. In unseren Rechnungen kennzeichnen wir Vektoren immer mit einem Pfeil. Man schreibt die Koordinaten eines Vektors senkrecht übereinander, im Gegensatz zur Koordinatenschreibweise von Punkten. Die Länge eines Vektors, auch Betrag genannt, errechnet man durch den Satz des Pythagoras. Der Vektor

$$\vec{v} = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$$

zeigt beispielsweise 3 Einheiten in x-Richtung und 4 Einheiten in y-Richtung. Sein Betrag ist

$$|\vec{v}| = \sqrt{3^2 + 4^2} = 5.$$

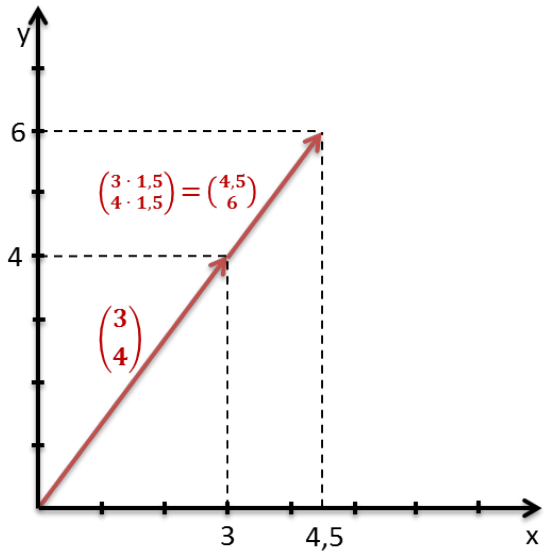
Wie kann man mit Vektoren rechnen?



Addition von Vektoren

Vektoren können wie Skalare miteinander addiert und subtrahiert werden, indem man die einzelnen Komponenten jeweils addiert bzw. subtrahiert. Jedoch kann man keine Vektoren mit Skalaren addieren oder subtrahieren. Ein Beispiel:

$$\begin{pmatrix} 3 \\ 4 \end{pmatrix} + \begin{pmatrix} 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 5 \\ 5 \end{pmatrix}$$



Multiplikation eines Vektors mit einem Skalar

Vektoren können mit Skalaren multipliziert werden, indem man jede Komponente des Vektors mit dem Skalar multipliziert. Zur Division multipliziert man jede Komponente mit dem Kehrwert des Divisors. Dadurch werden die Vektoren gestreckt oder gestaucht. Multipliziert man einen Vektor beispielsweise mit dem Skalar 1,5, dann ist der resultierende Vektor eineinhalbmal so lang, zeigt aber noch in dieselbe Richtung wie der ursprüngliche Vektor. Ein Beispiel:

$$1{,}5 \cdot \begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 1{,}5 \cdot 3 \\ 1{,}5 \cdot 4 \end{pmatrix} = \begin{pmatrix} 4{,}5 \\ 6 \end{pmatrix}$$

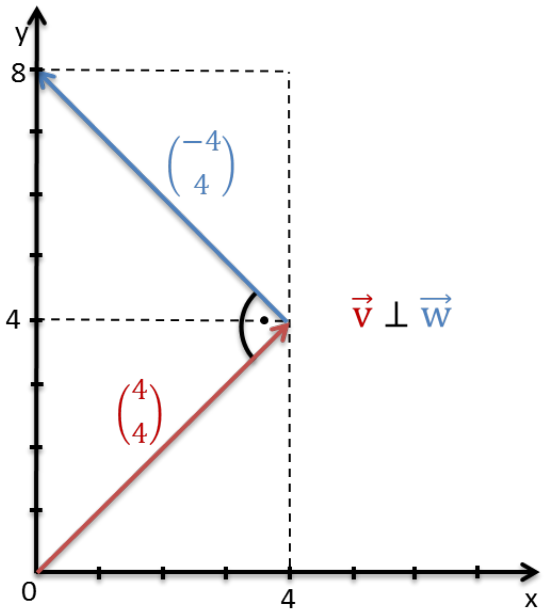
Wenn wir jetzt Vektoren miteinander multiplizieren wollen – was wir in der späteren Rechnung brauchen – definieren wir uns das Skalarprodukt. Hierbei werden die Werte jeweils einer Dimension miteinander multipliziert und am Ende alle Ergebnisse addiert:

$$\vec{v} \circ \vec{w} = \begin{pmatrix} v_x \\ v_y \end{pmatrix} \circ \begin{pmatrix} w_x \\ w_y \end{pmatrix} = v_x \cdot w_x + v_y \cdot w_y$$

Das Skalarprodukt wird durch einen „Kringel“ $\circ$ gekennzeichnet. Der Name Skalarprodukt kommt daher, dass das Ergebnis dieser Multiplikation ein Skalar ist. Ein Beispiel:

$$\begin{pmatrix} 4 \\ 4 \end{pmatrix} \circ \begin{pmatrix} -4 \\ 4 \end{pmatrix} = 4 \cdot (-4) + 4 \cdot 4 = -16 + 16 = 0$$

Hier sind wir gleich bei einem interessanten Spezialfall: Das Ergebnis des Skalarprodukts ist genau dann null, wenn die beiden Vektoren senkrecht aufeinander stehen.



Skalarprodukt zweier Vektoren

Den Winkel α zwischen den Vektoren $\vec{v}$ und $\vec{w}$ kann man ebenfalls mit Hilfe des Skalarprodukts errechnen. Die Formel hierfür lautet:

$$\cos\alpha = \frac{\vec{v} \circ \vec{w}}{|\vec{v}| \cdot |\vec{w}|}$$

Ein letztes Beispiel:

$$\cos\alpha = \frac{\begin{pmatrix} 3 \\ 4 \end{pmatrix} \circ \begin{pmatrix} 4 \\ 3 \end{pmatrix}}{5 \cdot 5} = \frac{3 \cdot 4 + 4 \cdot 3}{25} = \frac{24}{25}$$

$$\Rightarrow \alpha \approx 16{,}3°$$

Mit Hilfe dieser Formel können wir nun berechnen, in welchem Winkel zwei kollidierende Billardkugeln aufeinandertreffen.

Physikalische Grundlagen

Verena Winterhalter

Kinematik

Den Einstieg in die Physik, den wir für unser Billardprogramm brauchten, war die Kinematik – die Lehre der Bewegungen. Zunächst haben wir uns mit drei Formen der Bewegung beschäftigt, die so auch im Billard vorkommen: Mit dem Stillstand nur kurz, danach kam die gleichförmige Bewegung, also eine Bewegung mit gleichbleibender Geschwindigkeit. Bei allen Formen der Bewegung lässt sich die Durchschnittsgeschwindigkeit $\bar{v}$ aus der zurückgelegten Strecke Δs und der vergangenen Zeit Δt berechnen:

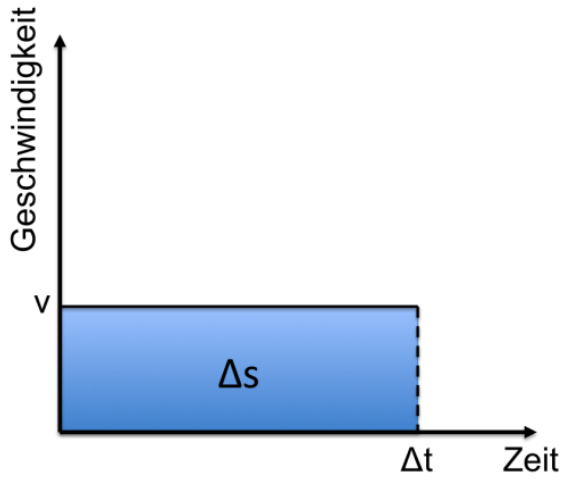
$$\bar{v} = \frac{\Delta s}{\Delta t}$$

Bei einer gleichförmigen Bewegung ist die Geschwindigkeit v konstant und es gilt:

$$v = \frac{\Delta s}{\Delta t}$$

Wenn man eine gleichförmige Bewegung in ein Geschwindigkeits-Zeit-Diagramm einzeichnet, stellt man fest, dass die zurückgelegte Strecke genau der Fläche unter dem Graphen entspricht.
Für den Flächeninhalt gilt:

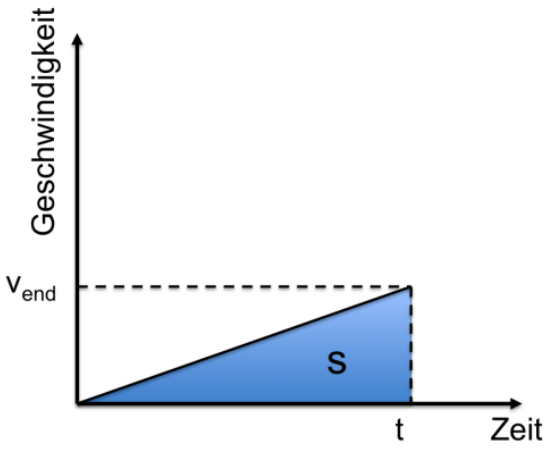
$$\Delta s = v \cdot \Delta t$$



Die dritte Bewegungsform, die wir uns angeschaut haben, war die gleichmäßig beschleunigte Bewegung. Sie wird durch die Beschleunigung a beschrieben. Eine Beschleunigung ist eine Geschwindigkeitsänderung pro Zeit. Dabei gilt:

$$a = \frac{\Delta v}{\Delta t} \tag{1}$$

Die bei einer beschleunigten Bewegung zurückgelegte Strecke Δs entspricht ebenfalls der Fläche unter dem Graphen im Geschwindigkeits-Zeit-Diagramm.



Da diese ein Dreieck beschreibt, lässt sich der Flächeninhalt folgendermaßen berechnen:

$$s = \frac{1}{2} v_{end} \cdot t.$$

Für die Endgeschwindigkeit v_{end} gilt:

$$v_{end} = a \cdot t$$

Man kann folgern:

$$s = \frac{1}{2} a t^2$$

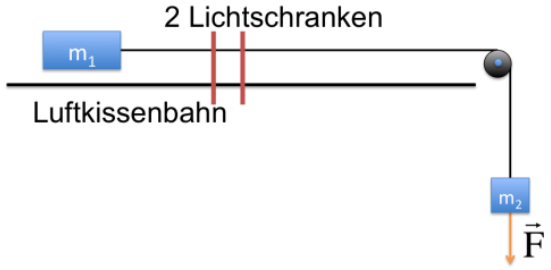
Wenn a größer wird, gewinnt das Objekt schneller an Geschwindigkeit. Ein negatives a beschreibt eine abgebremste Bewegung. Die Beschleunigung a hat die Einheit $\frac{\mathrm{m}}{\mathrm{s}^2}$.

Dynamik

Da zwei Kugeln, die aufeinander prallen, ihre Bewegungsrichtungen und ihre Geschwindigkeiten ändern, haben wir uns auch mit der Dynamik beschäftigt – der Lehre der Wirkung von

Kräften. Bewegungsänderungen von Körpern werden nämlich durch Kräfte verursacht.

Eine gleichmäßig beschleunigte Bewegung im Alltag ist der freie Fall. Hier wirkt die Schwerkraft. Zum freien Fall haben wir einen Versuch mit einer Luftkissenbahn aufgebaut: Auf einer Bahn liegt ein Wagen, den man mit Gewichten beschweren kann. Aus der Bahn strömt Luft aus, sodass der Wagen schwebt und somit fast keine Reibung zwischen der Bahn und dem Wagen wirkt. An dem Wagen ist durch eine Schnur und eine Umlenkrolle am Ende der Bahn ein Gewicht befestigt, das durch Loslassen des Wagens frei fällt und gleichzeitig den Wagen mit sich zieht. Mit Hilfe zweier Lichtschranken haben wir die Zeit für das Zurücklegen einer bestimmten Wegstrecke gemessen.



Damit konnten wir die Durchschnittsgeschwindigkeit des Wagens zwischen den Lichtschranken bestimmen. Da wir auch die Zeit gemessen haben, die der Wagen gebraucht hat, um die Lichtschranken zu erreichen, konnten wir mit Hilfe von Gleichung (1) die Beschleunigung des Wagens ermitteln. Durch diese Messungen haben wir bestimmt, dass die beschleunigende Kraft F proportional zur Beschleunigung a ist. Außerdem haben wir festgestellt, dass a umgekehrt proportional zur Gesamtmasse m ist. Demnach ist a proportional zu $\frac{F}{m}$. Historisch bedingt wurde der Proportionalitätsfaktor auf 1 gesetzt, sodass gilt:

$$F = m \cdot a$$

Man sagt auch: Kräfte führen Impulsänderungen herbei. Der Impuls p ist dabei definiert als

$$p = m \cdot v \tag{2}$$

woraus

$$F = \frac{\Delta p}{\Delta t}$$

folgt. Um Kollisionen von Billardkugeln korrekt zu berechnen, ist neben dem Impuls noch eine weitere Größe wichtig: Die Energie, die eine Kugel hat, wenn sie sich mit einer gewissen Geschwindigkeit v bewegt. Dazu stellen wir uns vor, die Kugel befände sich zunächst in Ruhe und würde dann durch eine Kraft F über eine gewisse Strecke s beschleunigt.
Energie ist in der Physik definiert als Produkt aus Kraft und Weg. Daher ändert sich die kinetische Energie um

$$E = F \cdot s.$$

Daraus folgt

$$E = ma \cdot \frac{1}{2}at^2 = \frac{1}{2}m(at)^2 = \frac{1}{2}mv^2.$$

Dies kann man wegen Gleichung (2) auch als

$$E = \frac{p^2}{2m}$$

schreiben.

Kollisionen

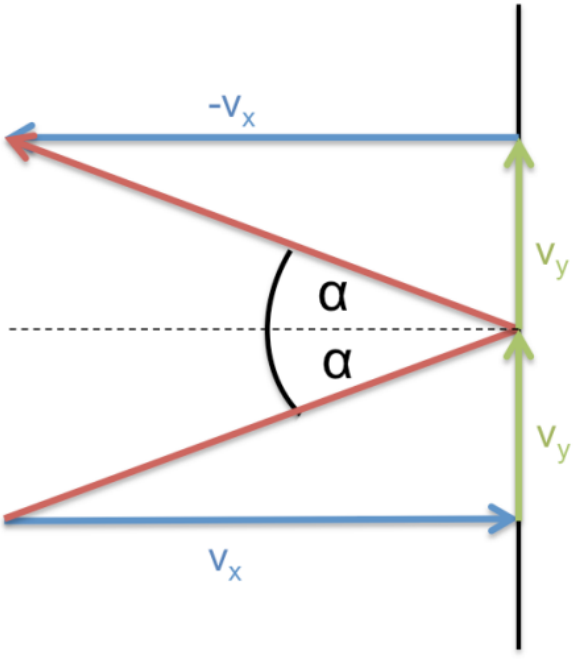
Tatjana Krenbauer

Wann kollidiert eine Kugel mit der Bande?

Da die Banden unseres Billardspiels entlang der Koordinatenachsen verlaufen, findet die Kollision einer Kugel mit einer Bande immer dann statt, wenn die Koordinaten des Mittelpunkts der Kugel plus bzw. minus den Radius der Kugel den Koordinaten der Bande entsprechen, oder leichter ausgedrückt, wenn die Kugel die Bande berührt.

Was passiert bei der Kollision?

Wenn eine Billardkugel auf eine Bande stößt, sagen die, die in Physik aufgepasst haben: „Einfallswinkel = Ausfallswinkel“. Das ist auch vollkommen richtig, wenn man davon ausgeht, dass es sich bei dem Zusammenstoß um einen elastischen Stoß handelt, also keine Energie durch Verformung, Geräusch, Reibung etc. verloren geht. Warum aber der Satz gilt, kann man nur vollständig erklären, wenn man bei dem Thema *Vektoren* gut aufgepasst hat. Denn die Geschwindigkeit $\vec{v}$, mit der unsere Kugel die Bande trifft, ist ein Vektor und lässt sich dementsprechend in eine Komponente, die parallel zur Bande und eine, die senkrecht zur Bande verläuft, zerlegen. Da eine Änderung nur in die Richtung stattfindet, in die eine Kraft wirkt, müssen wir nur die Komponente ändern, welche senkrecht zur Bande ist. Wir ändern sie, indem wir ihr Vorzeichen umkehren, wodurch die Geschwindigkeit genau in die entgegengesetzte Richtung verläuft. Wenn man das Ganze aufzeichnet, kann man sehen, dass der Satz *Einfallswinkel = Ausfallswinkel* wirklich gilt.



Wann kollidieren Kugeln miteinander?

Diese Frage ist etwas schwerer zu beantworten als die, wann eine Kugel gegen eine Bande stößt. Vor allem, wenn man die Kollisionen programmieren möchte. Das liegt daran, dass wir beim Programmieren alles in ein Koordinatensystem zeichnen, wodurch logischerweise Strecken, die nur in die x- oder nur in die y-Richtung verlaufen, einfacher zu berechnen sind, als Schrägen, bei denen die Steigung eine Rolle spielt. Dass die Kollision zweier Kugeln auch nur dann stattfindet, wenn sie sich berühren, ist klar. Und dass die Kugeln sich berühren, sobald der Abstand der Mittelpunkte der Kugeln d gleich der Summe ihrer Radien r ist, ebenfalls. Folglich kann man, wenn beide Kugeln den gleichen Radius haben, sagen, dass sie sich berühren, falls

$$d = 2r$$

gilt. Im Abschnitt zum Satz des Pythagoras haben wir ja bereits kennengelernt, wie man d berechnet:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Sollte d kleiner oder gleich werden als der doppelte Radius, liegt eine Berührung vor.

Was passiert bei der Kollision?

Jetzt, da wir wissen, wann es zu einer Kollision zwischen zwei Kugeln kommt, müssen wir nur noch wissen, welche physikalischen Gesetze gelten, wenn zwei Kugeln aneinander stoßen, um die Kollision dann auch realistisch programmieren zu können. Wenn man aufmerksam beobachtet, wie zwei Kugeln miteinander kollidieren, dann stellt man fest, dass der Gesamtimpuls $\vec{p}$ erhalten bleibt, das heißt die Summe der beiden Produkte von Masse und Geschwindigkeit der beiden Kugeln bleibt konstant. Kurz gesagt, es gilt der Impulserhaltungssatz:

$$\vec{p_1} + \vec{p_2} = \vec{p_1}' + \vec{p_2}'$$

$$m_1 \cdot \vec{v_1} + m_2 \cdot \vec{v_2} = m_1 \cdot \vec{v_1}' + m_2 \cdot \vec{v_2}'$$

Hierbei stehen die Größen mit einem Strich für die Impulse bzw. Geschwindigkeiten nach der Kollision. Außerdem gilt auch der Energieerhaltungssatz, denn wir gehen wie bei der Kugel-Banden-Kollision davon aus, dass beim

Stoß keine Energie verloren geht. Der Energieerhaltungssatz lässt sich auf verschiedene Weisen formulieren:

$$E_1 + E_2 = E_1' + E_2'$$
$$\frac{|\vec{p_1}|^2}{2m_1} + \frac{|\vec{p_2}|^2}{2m_2} = \frac{|\vec{p_1}'|^2}{2m_1} + \frac{|\vec{p_2}'|^2}{2m_2} \quad (3)$$

Es folgt eine Herleitung der Formel, wie viel schneller oder langsamer die einzelnen Kugeln nach einer Kollision miteinander sind. Für diese haben wir einen langen Nachmittag im Sommer benötigt:

Die Änderung des Impulses der ersten Kugel nennen wir $\Delta\vec{p}$. Wie bei der Kollision mit der Bande findet die Impulsänderung nur in der Richtung statt, in der eine Kraft wirkt, also in Richtung der Verbindungslinie der beiden Kugeln.

$$\Delta\vec{p} = k \cdot (\vec{x_1} - \vec{x_2})_0 = k \cdot \frac{\vec{x_1} - \vec{x_2}}{|\vec{x_1} - \vec{x_2}|} = k \cdot \vec{d_0} \quad (4)$$

k ist die Stärke der Impulsänderung – die wollen wir berechnen – und $\vec{d_0}$ beschreibt die Richtung der Impulsänderung – die wissen wir schon. $\vec{d_0}$ hat immer die Länge 1. Dies wird durch die tiefstehende 0 verdeutlicht.
Es gilt:

$$|\vec{d_0}| = 1 \quad (5)$$

Multipliziert man Gleichung (3) mit 2 auf beiden Seiten, so erhält man:

$$\frac{|\vec{p_1}|^2}{m_1} + \frac{|\vec{p_2}|^2}{m_2} = \frac{|\vec{p_1}'|^2}{m_1} + \frac{|\vec{p_2}'|^2}{m_2}.$$

Weiterhin gilt aber auch der Impulserhaltungssatz. Daher muss das, was die erste Kugel an Impuls ($\Delta\vec{p}$) gewinnt, von der zweiten Kugel abgegeben werden. Wenn wir den Impulserhaltungssatz mit einbeziehen, folgt:

$$\frac{|\vec{p_1}|^2}{m_1} + \frac{|\vec{p_2}|^2}{m_2} = \frac{|\vec{p_1} + \Delta\vec{p}|^2}{m_1} + \frac{|\vec{p_2} - \Delta\vec{p}|^2}{m_2} \quad (6)$$

Den Term $|\vec{p_1} + \Delta\vec{p}|^2$ können wir ausrechnen:

$$\begin{aligned} |\vec{p_1} + \Delta\vec{p}|^2 &= (\vec{p_{1x}} + \Delta\vec{p_x})^2 + (\vec{p_{1y}} + \Delta\vec{p_y})^2 \\ &= p_{1x}^2 + 2p_{1x} \cdot \Delta p_x + \Delta p_x^2 \\ &\quad + \; p_{1y}^2 + 2p_{1y} \cdot \Delta p_y + \Delta p_y^2 \\ &= |\vec{p_1}|^2 + 2\vec{p_1} \circ \Delta\vec{p} + |\Delta\vec{p}|^2 \end{aligned}$$

Analog gilt:

$$|\vec{p_2} - \Delta\vec{p}|^2 = |\vec{p_2}|^2 - 2\vec{p_2} \circ \Delta\vec{p} + |\Delta\vec{p}|^2$$

Diese Ergebnisse können wir nun in Gleichung (6) einsetzen:

$$\begin{aligned} \frac{|\vec{p_1}|^2}{m_1} + \frac{|\vec{p_2}|^2}{m_2} &= \frac{|\vec{p_1}|^2}{m_1} + 2\frac{\vec{p_1} \circ \Delta\vec{p}}{m_1} + \frac{|\Delta\vec{p}|^2}{m_1} \\ &\quad + \frac{|\vec{p_2}|^2}{m_2} - 2\frac{\vec{p_2} \circ \Delta\vec{p}}{m_2} + \frac{|\Delta\vec{p}|^2}{m_2} \\ 0 &= |\Delta\vec{p}|^2 \cdot \left(\frac{1}{m_1} + \frac{1}{m_2}\right) \\ &\quad + 2 \cdot \Delta\vec{p} \circ (\vec{v_1} - \vec{v_2}) \end{aligned}$$

Für $\Delta\vec{p}$ setzen wir jetzt die Formel aus Gleichung (4) ein:

$$0 = k^2 |\vec{d_0}|^2 \cdot \left(\frac{1}{m_1} + \frac{1}{m_2}\right) + 2k \cdot \vec{d_0} \circ (\vec{v_1} - \vec{v_2})$$

Nun teilen wir durch k:

$$0 = k|\vec{d_0}|^2 \cdot \left(\frac{1}{m_1} + \frac{1}{m_2}\right) + 2 \cdot \vec{d_0} \circ (\vec{v_1} - \vec{v_2})$$

$|\vec{d_0}|^2$ ist wegen Gleichung (5) gleich 1.
Da es ja unser Ziel ist, k zu berechnen, lösen wir die Gleichung nach k auf:

$$k = \frac{2\vec{d_0} \circ \vec{v_2} - 2\vec{d_0} \circ \vec{v_1}}{\frac{1}{m_1} + \frac{1}{m_2}} \quad (7)$$

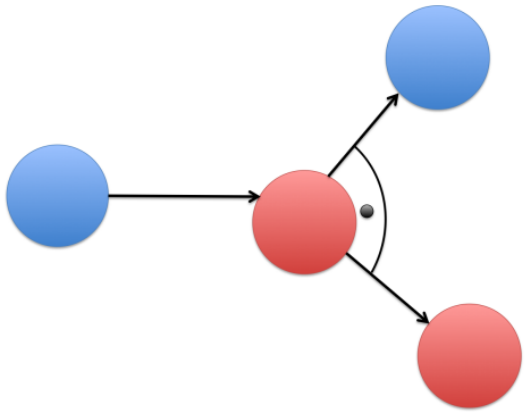
Für den Computer muss man die Formel etwas umschreiben, um sie programmieren zu können:

$$\begin{pmatrix} p_{1x} \\ p_{1y} \end{pmatrix}' = \begin{pmatrix} p_{1x} \\ p_{1y} \end{pmatrix} + k \cdot \begin{pmatrix} d_{0x} \\ d_{0y} \end{pmatrix}$$

Spezialfälle

Wenn zwei Kugeln mit unterschiedlichen Geschwindigkeiten aber gleicher Masse frontal aufeinander stoßen, dann tauschen sie ihre Geschwindigkeiten.

Wenn die Kugeln jedoch nicht exakt frontal aufeinander stoßen, dann laufen sie immer im rechten Winkel auseinander, wieder unter der Voraussetzung, dass sie die gleiche Masse haben.



Stoß zweier Kugeln gleicher Masse

Reibung

Verena Winterhalter

Wenn man eine Billardkugel anspielt, kann diese gegen die Bande oder andere Kugeln stoßen, irgendwann bleiben aber alle Kugeln stehen. Dies liegt an der Reibung, die zwischen den Kugeln und dem Tischtuch wirkt.

Reibung entsteht, da weder Billardtuch noch Billardkugeln eine absolut glatte Oberfläche besitzen. Wenn man sich die Oberflächen sehr genau ansieht, sind sie sehr unregelmäßig und ihre Hebungen und Senkungen greifen ineinander. Wenn man diese zwei Flächen gegeneinander verschieben möchte, muss man Kraft aufwenden, um diese „Verzahnung“ zu lösen, wodurch Energie verloren geht. Diesen Energieverlust bezeichnet man als Reibung.

Es gibt drei verschiedene Arten von Reibung: Haftreibung, Gleitreibung und Rollreibung.
Die Haftreibung tritt direkt beim Queue-Stoß gegen die Kugel auf und ist die Kraft, die man braucht, um die Kugel von ihrer Ruheposition auf dem Billardtisch weg zu bewegen. Die Gleitreibung tritt kurz nach dem Stoß ein, denn erst gleitet die Kugel kurz über den Billardtisch, bevor sie anfängt zu rollen. Sobald die Kugel rollt, muss die Rollreibung beachtet werden. Diese wirkt so lange, bis die Kugel anhält. Die Kraft, die man benötigt, um die jeweilige Reibung zu überwinden, ist bei der Haftreibung am größten und bei der Rollreibung am kleinsten. Da die Haft- und Gleitreibung jedoch viel kürzer wirken als die Rollreibung, haben wir beschlossen, den Schwerpunkt auf das Einbinden der Rollreibung zu legen. Damit das Ganze realistisch wirkt, haben wir uns überlegt, nicht einfach irgendeinen Wert für die Reibung zu verwenden, sondern den realen Wert durch einen entsprechenden Versuch zu ermitteln.

Der Versuch musste reproduzierbar sein, damit wir mit dem Durchschnittswert mehrerer Messungen einen möglichst genauen Wert berechnen konnten. Der Versuchsaufbau sah so aus, dass wir mit einem Stativ eine Rampe gebaut haben, von deren oberen Ende die Billardkugel losgerollt ist. Auf einem Tisch hatten wir ein Billardtuch gespannt, über welches die Kugeln dann nach dem Ende der Rampe weiter gerollt sind. Damit durch den Übergang zwischen Rampe und Billardtuch keine Messfehler entstehen, haben wir die Zeit erst ab einer mit Kreide auf dem Billardtuch markierten Start-Linie gemessen. Gemessen wurde die Zeit, die die Kugel von der Start-Linie bis zum Stillstand gebraucht hat. Die Stelle, an der die Kugel anhält, wurde ebenfalls mit Kreide markiert. Dann haben wir den Abstand zwischen der Stillstand-Markierung und der Start-Linie gemessen. Mit diesen zwei Werten konnten wir den Rollreibungskoeffizienten einer Billardkugel auf einem Billardtuch berechnen. Um mögliche Messfehler auszugleichen, wurde die Zeit dabei immer von fünf Personen gleichzeitig gestoppt. Außerdem haben wir die Höhe der Rampe und somit die Anfangsgeschwindigkeit variiert. Wir haben festgestellt, dass die Reibung unabhängig von der Geschwindigkeit ist. Den berechneten Wert von $-5{,}8\,\mathrm{cm/s^2}$ haben wir dann entsprechend in unser Programm übertragen. Die Einheit der Rollreibung $\mathrm{cm/s^2}$ bedeutet dabei, dass die Kugel pro Sekunde um $5{,}8\,\mathrm{cm/s}$ langsamer wird.

Spin

David Elsing

Als Spin (engl. Drall) wird beim Billard die Rotation der Kugeln bezeichnet. Er ist insbesondere bei der weißen Kugel wichtig, weil damit besondere Stöße durchgeführt werden können.

Um mit Drehbewegungen rechnen zu können, braucht man erst einmal passende Größen: Die Drehgeschwindigkeit kann man nicht einfach in Meter pro Sekunde angeben, denn Punkte, die sich weiter weg von der Drehachse befinden, bewegen sich schneller als Punkte, die näher an der Achse liegen. Deswegen gibt es analog zu den Größen bei „normalen Bewegungen“ Größen für Drehbewegungen. Um die Geschwindigkeit der Drehung anzugeben, braucht man, wie bei „normalen“ Geschwindigkeiten auch, zwei Größen: Die Zeit und die „Strecke“, die bei Drehbewegungen durch einen Winkel angegeben wird. Dabei ist es üblich, den Winkel im Bogenmaß (Einheit Radiant) anzugeben, was das Rechnen viel einfacher macht. Das Formelzeichen dafür ist φ. Analog zur „normalen“ Geschwindigkeit gibt es die Winkelgeschwindigkeit mit dem Formelzeichen ω, die sich durch Winkel pro Zeit bzw. Radiant pro Sekunde berechnen lässt:

$$\omega = \frac{\varphi}{t}$$

In unserem Billardspiel haben wir nur die Rollbewegung der Kugeln in Bewegungsrichtung eingebaut (also war die Drehachse immer senkrecht zur Bewegungsrichtung). Dabei wird auch nicht beachtet, dass die Kugel beim Stoß eigentlich erst gleitet, bevor sie anfängt zu rollen. Sie dreht sich also bei gleicher Geschwindigkeit immer gleich schnell.

Und wie schnell dreht sich eine Kugel? Dazu muss man wissen, dass die zurückgelegte Strecke dem zurückgelegten Umfang, also $\varphi \cdot r$, entspricht. Bei einer ganzen Umdrehung legt die Kugel also einen Weg von $2\pi r$ zurück – ihren Umfang. Die Winkelgeschwindigkeit in unserem Billardspiel entspricht also

$$\omega = \frac{\varphi}{t} = \frac{s}{t \cdot r} = \frac{v}{r}.$$

Obwohl wir nur diese Rollbewegung implementiert haben, haben wir uns trotzdem noch etwas näher mit dem Spin beschäftigt:

Analog zur Masse m gibt es bei Drehbewegungen nämlich noch das Trägheitsmoment I, eine Art „Drehmasse“. Das Problem dabei besteht darin, dass es davon abhängt, welchen Abstand die Masse zur Rotationsachse hat. Ein sehr anschauliches Beispiel ist der Pirouetteneffekt, also dass es schwerer wird, sich zu drehen, wenn man Masse von der Drehachse entfernt (also z. B. die Arme ausstreckt). Bei Kugeln mit homogener (gleichmäßiger) Massenverteilung ist das Trägheitsmoment – also die Drehmasse – bei Drehachsen durch den Mittelpunkt der Kugel immer gleich und lässt sich mithilfe der Masse m und des Radius r so berechnen:

$$I = \frac{2}{5} m r^2$$

Aus den oben definierten Größen lassen sich noch einige andere Größen berechnen:

Die Winkelbeschleunigung $\alpha = \frac{\omega}{t}$ ist analog zur „normalen“ Beschleunigung die Änderung der Drehgeschwindigkeit pro Zeit.

Der Drehimpuls L und das Drehmoment D entsprechen den Größen Impuls und Kraft.

Der Drehimpuls lässt sich schreiben als $L = I \cdot \omega$ (ganz analog zum „normalen“ Impuls $p = m \cdot v$), wodurch sich auch der Pirouetteneffekt erklären lässt: Das Trägheitsmoment ändert sich, der Drehimpuls bleibt aber gleich, also muss sich die Winkelgeschwindigkeit ändern.

Das Drehmoment entspricht dem Produkt aus der angreifenden Kraft und ihrem Abstand zur Drehachse: $D = F \cdot s$. Dies ist beim Hebelgesetz wichtig, weil sich mit größerem Abstand von der Drehachse der Hebel vergrößert.

Zum Hebelgesetz: Wippe im Gleichgewicht

Beispiel:
Im Bild ist die Wippe ausbalanciert, weil auf beiden Seiten das gleiche Drehmoment wirkt.

$$F_1 = m_1 g = 100\,\text{kg} \cdot 9.81\,\frac{\text{N}}{\text{kg}} = 981\,\text{N}$$

$$D_1 = F_1 s_1 = 981\,\text{N} \cdot 10\,\text{cm} = 98,1\,\text{Nm}$$

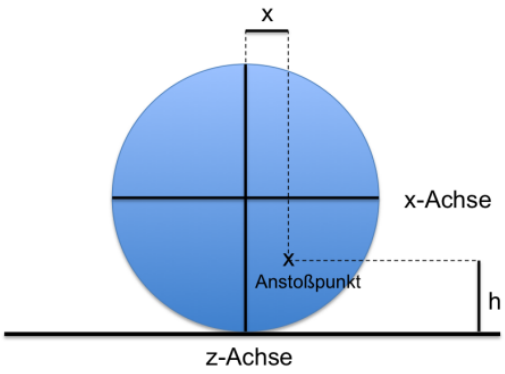
$$F_2 = m_2 g = 5\,\text{kg} \cdot 9.81\,\frac{\text{N}}{\text{kg}} = 49,05\,\text{N}$$

$$D_2 = F_2 s_2 = 49,05\,\text{N} \cdot 200\,\text{cm} = 98,1\,\text{Nm}$$

$$\Rightarrow D_1 = D_2$$

Jetzt muss man natürlich berechnen, wie sich der Spin durch die Reibung auf Richtung und Größe der Geschwindigkeit der Kugel auswirkt und wie man der Kugel mit dem Queue Spin verleiht.

Führt man den Queue parallel zum Tisch, so kann beim Stoß der Kugel nur Spin um Achsen geben, die senkrecht auf der Bewegungsrichtung stehen.



Eine Drehung um die z-Achse nach dem Stoß entspricht

$$\omega_z = \frac{F \cdot x}{I} \cdot t_{eff} = \frac{5Fx}{2mr^2} \cdot t_{eff}.$$

F ist dabei die Kraft des Anstoßes, und t_{eff} ist die Zeit, in der der Queue die Kugel berührt und beschleunigt.

Die Drehung um die x-Achse lässt sich so berechnen:

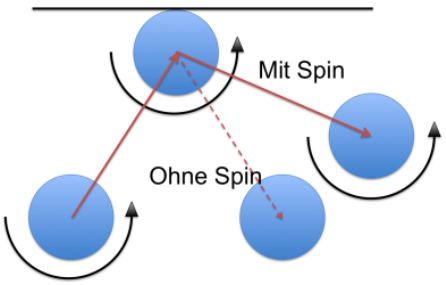
$$\omega_x = \frac{-5F \cdot (r-h)}{2mr^2}$$

Die Geschwindigkeit der Kugel nach dem Stoß entspricht

$$v = \frac{F}{m} \cdot t_{eff}.$$

Während die Kugel rollt, passieren zwei Dinge: Der Spin wird durch die Reibung zum Tisch geändert. Der Spin ändert aber durch Reibung auch die Geschwindigkeit der Kugel. Das passiert so lange, bis sich die Kugel so schnell dreht, dass sie zu rollen beginnt.

Die Bewegung der Kugel durch den Spin um die z-Achse bleibt so gut wie unverändert, ändert an der Bande jedoch den Reflexionswinkel:



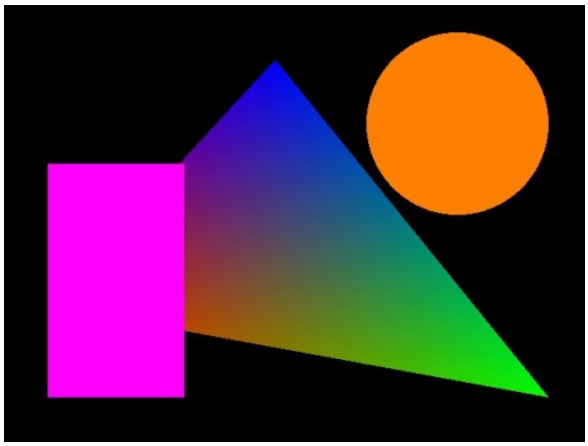
Grafik mit OpenGL

Lukas Dippon

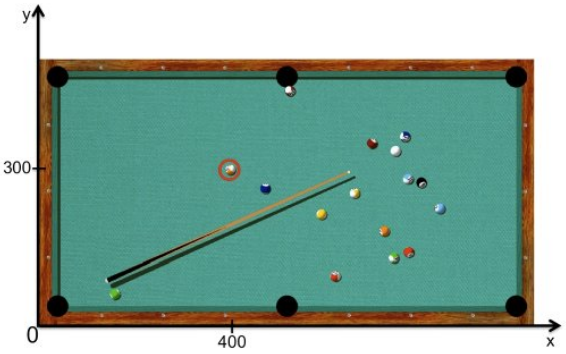
Unsere ersten Programme konnten bereits rechnen und Texte ausgeben, auch einfache Formen hatten wir schon gezeichnet. Dafür benutzten wir die OpenGL-Grafikbibliothek. Diese bietet eine große Vielfalt an Befehlen, mit denen wir relativ einfach Rechtecke, Dreiecke und dank Daniel, der uns noch ein paar zusätzliche Befehle zur Verfügung stellte, auch Kreise zeichnen konnten.

Unser erstes Spiel bestand nur aus einem Tisch, auf dem eine Kugel hin und her rollte. Dann kamen mehr Kugeln, ein Queue und vieles mehr dazu. Doch wie genau funktioniert das Zeichnen eigentlich?

Wir haben ein Fenster, über welches wir ein zweidimensionales Koordinatensystem legen, wie man es aus der Schule kennt. Eine Einheit auf den Koordinatenachsen entspricht einem Pixel, also einem Bildpunkt des Monitors. Mit diesem Koordinatensystem konnten wir dem



Computer sagen, wo genau die Objekte gezeichnet werden sollen. Ein Dreieck definiert sich beispielsweise über die Koordinaten seiner drei Ecken, ein Kreis über seinen Mittelpunkt und über seinen Radius. So erhielten wir hübsche Bilder, die sich leider noch nicht bewegen konnten.



Unser Billardtisch im Koordinatensystem

Um zu zeigen, wie wir bewegte Bilder erzeugen können, gehen wir einmal von einer sehr vereinfachten Billard-Version aus, die nur aus einem Spielfeld, einer Kugel und einem Queue, mit dem wir die Kugel anspielen können, besteht. Um das zu bewerkstelligen, benutzten wir die idle-Funktion. Sie ist eine Liste von Befehlen, die immer dann ausgeführt wird, wenn der Computer ein Bild fertig gezeichnet hat.

```
void idle() {

  double t = diff_seconds();

  kugel.bewegen(t);
  kugel.kollision(tisch);
  queue.kollision(kugel);
  display();

}
```

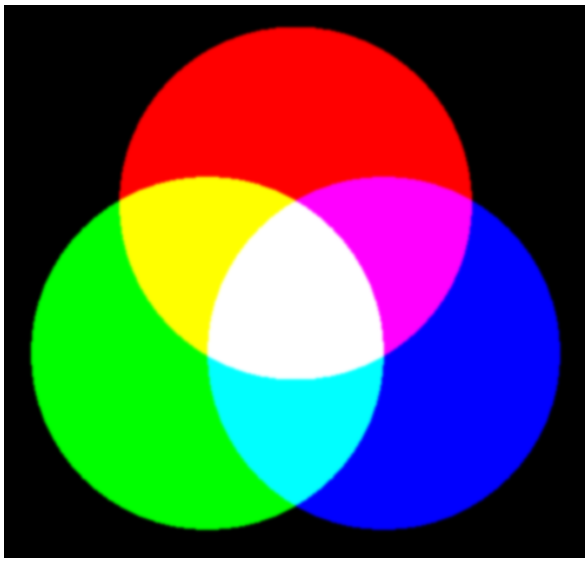
Da dieser Vorgang, je nach Rechenleistung des PCs und der momentanen Auslastung, unterschiedlich lange dauern kann, messen wir mit dem Befehl `diff_seconds` als allererstes die Zeit, die seit dem letzten Durchlauf vergangen ist und speichern sie in der Variable `t`. Dadurch können wir in der Funktion `kugel.bewegen` die Bewegungen der Kugeln physikalisch korrekt berechnen.
Als nächstes überprüft die Funktion `kugel.kollision`, ob die Kugel innerhalb dieses Vorgangs an die Bande geprallt ist und ändert gegebenenfalls ihre Richtung. Im nächsten Schritt wird noch überprüft, ob der Queue die ruhende Kugel anstößt.
Zuletzt wird die Funktion `display` aufgerufen, mit der das ganze Bild neu gezeichnet wird. Je nach Geschwindigkeit des verwendeten Computers werden fünfzig bis mehrere hundert Bilder pro Sekunde berechnet, sodass für unser Auge ein flüssiges Bild entsteht.

Das Farbsystem von OpenGL

Myriam Holl

OpenGL verwendet das sogenannte RGB-Farbsystem. Hierbei steht R für Rot, G für Grün und B für Blau. Aus diesen drei Farben lassen sich alle weiteren, beliebigen Farben mischen. Das RGB-Farbsystem ist ein sogenanntes additives Farbsystem (oder auch additive Farbsynthese oder physiologische Farbmischung). Auf dem folgenden Bild kann man zum Beispiel

erkennen, dass die Kombination von rotem, grünem und blauem Licht Weiß ergibt. Wenn nur Rot und Blau vorhanden sind, entsteht Magenta, aus Grün und Blau wird Cyan, Grün und Rot addieren sich zu Gelb. Schwarz entsteht dann, wenn keine Lichtfarben vorhanden sind.



Warum kann man aber überhaupt Farben als Summe von diesen drei Grundfarben darstellen? Die Antwort liegt im Bau unseres Auges. Beim menschlichen Auge unterscheidet man drei Farbrezeptortypen (drei Zapfentypen), den L-Typ (der Rotrezeptor), den M-Typ (der Grünrezeptor) und den S-Typ (der Blaurezeptor). Diese werden jeweils von rotem, grünem und blauem Licht gereizt. Die drei Farbrezeptoren ermöglichen zusammen mit den Stäbchen, die für die Unterscheidung von hell und dunkel zuständig sind, dem Auge das Sehen. Das Gehirn berechnet aus den jeweiligen Intensitäten der Grundfarben die resultierende Farbe. Wenn man jetzt jeden Farbrezeptortyp gezielt auf seine spezifische Farbe anspricht, kann man dem Gehirn jeden beliebigen Farbeindruck simulieren. Daher besteht ein Computermonitor im Prinzip aus vielen kleinen Lämpchen, die genau diese drei Grundfarben emittieren und damit beliebige Farben mischen können.

Der OpenGL-Befehl um eine Farbe zu setzen, lautet `glColor3f (1.0, 1.0 ,1.0)` .

Dabei gibt das Suffix `3f` an, dass drei Fließkomma-Werte an die Funktion übergeben werden müssen. Der erste Wert steht für die Intensität des Rotanteils im Licht, der zweite für den Grünanteil und der dritte für den Blauanteil. Diese Werte liegen immer zwischen 0,0 und 1,0. Je größer der Wert, desto heller das Licht und damit die Farbe. Das bedeutet, dass beispielsweise `glColor3f (1.0, 1.0, 1.0)` Weiß entspricht und `glColor3f (0.0, 0.0, 0.0)` Schwarz. Ein mittelhelles Grau erhält man zum Beispiel durch `glColor3f (0.5, 0.5, 0.5)`. Der Computer „addiert“ dann die drei Farbintensitäten und man erhält die entsprechende Farbe.

Da der Computer 256 Rottöne, 256 Grüntöne und 256 Blautöne verwendet, können somit $256 \cdot 256 \cdot 256 \approx 16{,}8$ Millionen verschiedene Farben auf dem Monitor dargestellt werden. Die unterschiedlichen Farbtöne liegen dabei so nah beisammen, dass das menschliche Auge gar nicht alle unterscheiden kann.

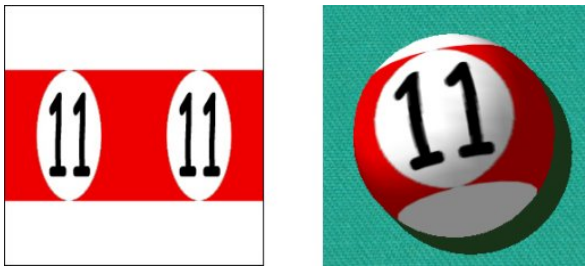
Texturen

Lukas Dippon

Bisher konnten wir den Computer einfache farbige Figuren zeichnen lassen, zum Beispiel Rechtecke, Kreise oder Dreiecke. Aber das war uns immer noch zu langweilig. Außerdem wäre es extrem mühsam, die Oberfläche des Spiels, zum Beispiel die des Tisches, mit lauter Rechtecken zu zeichnen und das Resultat wäre wahrscheinlich alles andere als zufriedenstellend. Deshalb haben wir uns mit einer Kamera auf die Suche gemacht und mehrere Bilder von Objekten und deren Oberflächen erstellt, die wir dann später in unser Spiel eingebaut haben. Dazu gehörten verschiedene Holzmuster, ein Billardtuch für den virtuellen Tisch, Edelsteine von Ohrringen (Diamanten) als Orientierungshilfen auf der Bande. Diese Bilder klebten wir dann ähnlich einer Tapete auf die zuvor einfarbigen Flächen unseres Spiels. Mit diesen Texturen konnten wir unser Spiel noch einmal beträchtlich aufpeppen. So sah nach etwas Programmierarbeit, in der wir die Texturen in das Programm einbauten, unser Spiel doch gleich deutlich realistischer aus. Dies war noch relativ einfach, denn die Texturen und die Flächen, die

wir bekleben wollten, waren zweidimensional, also flach.

Ein weiterer Verwendungszweck für die Texturen, die wir dieses Mal allerdings am Computer gestalteten, waren die Kugeln. Wie vorher bereits erwähnt, können unsere Kugeln im endgültigen Spiel rollen und gleiten nicht einfach über den Tisch. Um das zu verdeutlichen, haben wir ebenfalls Texturen verwendet, da man einer einfarbigen Kugel ja nicht ansieht, ob sie nun rollt oder nicht. Die Herausforderung bestand nun jedoch darin, die zweidimensionalen, flachen Texturen auf die dreidimensionale, gekrümmte Oberfläche der Kugel zu kleben.



Die Textur einer Kugel und die texturierte Kugel im Spiel

Allerdings gibt es praktischerweise in OpenGL bereits eine Funktion, die diese Aufgabe für uns übernommen hat. Ein weiteres Problem bestand darin, die Texturen so zu gestalten, dass sie nachher auf der Kugel realistisch aussehen, da sie bei dem Aufkleben verzerrt werden, ähnlich wie wenn man versucht die Erdkugel auf einer flachen Karte abzubilden. Also gestalteten wir ein Bild, das die Textur nach oben gestreckt darstellte. Diese Verzerrung wurde dann durch das Aufkleben wieder ausgeglichen.

Dreidimensionale Effekte

Sebastian Dallinger

Um das Spiel noch realistischer aussehen zu lassen, brauchten wir jedoch mehr als nur schöne Texturen. Um den Eindruck eines „echten“ Billardtisches zu vermitteln, ist alles in dreidimensionaler Optik gehalten. Bei den glatten Oberflächen, wie dem Spielfeld oder den Banden, mussten wir hierfür nichts speziell in unser Spiel einbauen. Grundsätzlich haben wir mit drei Methoden zur dreidimensionalen Darstellung gearbeitet:

Größenveränderung

Wenn eine Kugel in eines der Löcher fällt, entfernt sie sich vom Betrachter, der von oben auf das Spielfeld schaut – sie wird also kleiner:

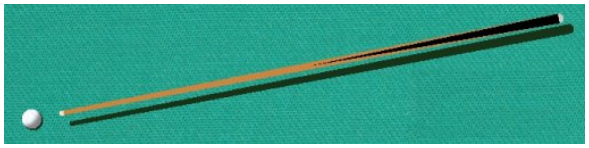
Umgesetzt wurde dieser Effekt durch die Formel:

$$r_{neu} = r_{alt} - 20 \cdot t$$

Hierbei bezeichnet r den Radius der Kugel und t die seit dem letzten Zeichnen des Bildes vergangene Zeit. Durch diese Formel wird also der Kugelradius um zwanzig Pixel pro Sekunde verringert, bis die Kugel komplett im Loch verschwunden ist.

Schatten

Ein einfaches, aber effektives Mittel um einen dreidimensionalen Eindruck zu erzeugen, ist das Zeichnen von Schatten. In unserem Programm hat jedes Objekt einen Schatten: Im Fall der Banden ein dunklerer Bereich direkt an der Bande, im Fall des Queues und der Kugeln eine dunkle Fläche, die sich mit diesen mitbewegt und jeweils immer etwas nach rechts unten versetzt ist.



Am Beispiel der Kugel sieht das wie folgt aus:

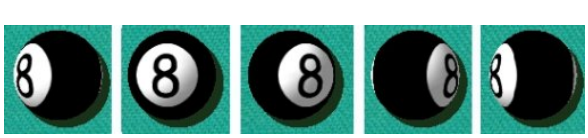
```
void Kugel::schatten()  {

  glColor3f(0.1, 0.2, 0.1);

  graphicsBall(x + radius / 12,
      y - radius / 12, radius);
}
```

Es wird ein dunkelgrüner Kreis gezeichnet, der gegenüber dem Kugelmittelpunkt um ein zwölftel des Radius nach rechts unten verschoben ist und den gleichen Radius wie die Kugel hat. Im Programm erscheint dieser Kreis als Schatten der Kugel.

Lichtverlauf und Textur bei Kugeln

Die größte Herausforderung bei der grafischen Entwicklung unseres Programms war, die Kugeln wirklich zum Rollen zu bringen. Glücklicherweise stellt uns OpenGL hier eine eigene Funktion zur Verfügung, die ausschließlich dem Zweck dient, dreidimensionale Kugeln zu zeichnen. Da das Kennenlernen, Verstehen und schließliche Einbauen dieser Funktion den Zeitrahmen wahrscheinlich um Wochen gesprengt hätte, wurde dieser Teil freundlicherweise von Daniel übernommen und grundlegend erklärt.

Zu jeder Kugel wird ihre aktuelle Drehung gespeichert, also wie weit sie sich in welche Richtung gedreht hat. Die der Kugel zugewiesene Textur wird entsprechend mitgedreht, wodurch der Eindruck einer rollenden Kugel entsteht:



Präsentationen

Ludwig Bald

Neben unserer Arbeit am Billardprogramm und unseren theoretischen Überlegungen in der Physik und Mathematik haben wir im Laufe der Akademie zwei Vorträge über unsere Kursinhalte gehalten.

Rotation

Bei der Rotation geht es einerseits darum, den anderen Kursen die Zwischenergebnisse nach der Hälfte der Akademie zu präsentieren, andererseits soll sie auch auf die Abschlusspräsentation vorbereiten. Dafür wurde jeder Kurs in vier Präsentationsgruppen mit je drei Leuten unterteilt, die dann jeweils den Teilnehmern der anderen Kurse vorgetragen haben und sich natürlich auch die Präsentationen der anderen Kurse anhören durften.

Zur Vorbereitung unserer Präsentationen haben wir uns als erstes einen groben Aufbau überlegt: Wir unterteilten die Präsentation in drei Teile – einen für die Arbeit mit dem Computer, einen für die Physik und einen für die Moderation sowie allgemeine Informationen zu unserem Spiel. Nachdem wir die Präsentationsgruppen gebildet hatten, setzten wir pro Gruppe einen Experten für jeden Teil ein. Diese setzten sich in Expertengruppen zusammen und entwickelten den jeweiligen Teil der Präsentation. Als nächstes hielten wir mehrere Testvorträge und bekamen nützliche Ratschläge von unseren Kursleitern und unserer Schülermentorin, wie man besser vorträgt. Das waren sowohl rhetorische Tricks, als auch eine Tabuliste von Ausdrücken wie beispielsweise „das ist ganz einfach“, „keine Ahnung“, und „äh“.

Bevor es an die Präsentationen ging, zeigte uns Alex als angeblicher Rhetorikbeauftragter seiner Universität am Mittwochmorgen, wie unsere Präsentationen nicht aussehen sollten. Nachdem Daniel und Alex uns am Donnerstag im Plenum einprägsam daran erinnert hatten, keine Silben zu verschlucken, starteten wir gut gelaunt und zuversichtlich in den Präsentationstag.

Alle Präsentationen verliefen gut und wurden von anderen Kursteilnehmern und -leitern als leicht verständlich und anschaulich bezeichnet. Ist doch klar, mit dem Rhetorikbeauftragten als Kursleiter!

Abschlusspräsentation

Am Ende der Akademie hatten unsere Familien und andere wichtige Gäste die Gelegenheit, sich unsere Kursarbeit der vergangenen zwei Wochen anzuschauen. Unsere dafür vorbereiteten Abschlusspräsentationen waren im Großen und Ganzen so aufgebaut wie die Vorträge zur Rotation, wurden jedoch auf das breiter gefächerte Publikum angepasst. Natürlich ergänzten wir sie um die in der zweiten Woche behandelten Themen. Das waren beispielsweise der Versuch zur Bestimmung des Reibungskoeffizienten oder der mathematische Hintergrund der rollenden Kugeln.

Auch die Abschlusspräsentationen verliefen super: Das Publikum im voll besetzten Raum hörte gebannt zu und zeigte sich begeistert von unseren Vorträgen. Nach den Vorträgen wurden natürlich noch offene Fragen beantwortet und anschließend durften die Zuhörer unser selbstprogrammiertes Billardspiel ausprobieren, was dazu führte, dass auch in den Pausen die Computer von vielen begeisterten Besuchern besetzt waren.

Am Ende des Tages waren wir alle mit den gehaltenen Vorträgen zufrieden und konnten uns auf den Abschlussabend freuen.

Ausgleichssport

Ludwig Bald

Natürlich haben sich nicht nur die Billardkugeln auf unseren Monitoren bewegt, sondern auch wir. Innerhalb und außerhalb des Kurses haben wir uns sportlich betätigt und dabei viel Spaß gehabt.

Das Sportfest

Die Erwartungen waren hoch gesteckt für unseren Kurs: Schon im letzten Jahr gewann der Physikkurs (allerdings mit anderen Kursleitern) das Sportfest. Um den Titel und die Ehre zu verteidigen, mussten wir die anderen Kurse in vielen verschiedenen Disziplinen schlagen, um nur einige zu nennen:

- *Autoziehen:* Beim Autoziehen ging es darum, einen VW-Bus quer über den ganzen Parkplatz zu ziehen. Keine leichte Aufgabe, wenn der eigene Kursleiter am Steuer sitzt und vergisst, das Lenkradschloss zu lösen.
- *Schwimmflossen:* Es galt einen Parcours als Staffel mit Schwimmflossen zu durchlaufen und dabei mit der sehr limitierten Auswahl an Schwimmflossen auszukommen.
- *Das laufende A:* Ziel dieser Aufgabe war es, Irina in einer A-förmigen Konstruktion ins Ziel zu bugsieren. Einziges Hilfsmittel waren vier am oberen Eck des As angebrachte Seile. Hier waren vor allem Koordination und Teamgeist gefragt.

- *Balance:* Man nehme eine Sportbank, stelle sich darauf, singe den Musik-KüA-Leitern einmal „Hänschen klein“ vor und ordne sich dann nach Anfangsbuchstaben des Nachnamens – ohne dabei herunterzufallen, versteht sich.
- *Das große Finale:* Sechs Kurse, sechs Eimer, sechs Schwämme, ein Ziel: Den eigenen Ei-

mer so schnell wie möglich mit Wasser zu füllen. Allerdings nur mit einem Schwamm, den man einmal pro Lauf aufsaugen lassen darf. Wir lagen ziemlich gut im Rennen, bis Sekunden vor dem Abpfiff aufgrund nassen Grases der Eimer einfach umfiel und halbleer lief.

Die Siegerehrung

Wir hatten die haarsträubenden Ergebnisse des Sportfests schon fast vergessen, als einige Tage später beim Bergfest zur Siegerehrung aufgerufen wurde. Eigentlich hatten wir damit gerechnet, dass uns zumindest der Medizinkurs überholt hat, vor allem nach diesem verpatzten Finale. Insofern war die Freude groß, als dieser schon für den zweiten Platz auf die Bühne gerufen wurde und somit klar wurde, dass wir gewonnen hatten. Wir gewannen einen Korb voller Obst und Süßigkeiten, damit wir auch weiterhin viele Sportfeste gewinnen.

Der Physikkurs im nächsten Jahr hat den Titel wieder zu verteidigen!

Unser kursinternes Billardturnier

Man kann kein Billardspiel programmieren, ohne einmal richtig Billard gespielt zu haben. Deswegen sammelten wir an einem Nachmittag lang Praxiserfahrung, indem wir mit zufällig ausgewählten Teams ein Billardturnier austrugen. Dieses gewannen Frank und Lukas. Sie triumphierten nach dem Abendessen sogar mit 2 zu 1 gewonnen Spielen gegen unsere Kursleiter Daniel und Alex.

Exkursion nach Heidelberg

Lukas Dippon

Fast alle Kurse (außer TheoPrax) gingen am zweiten Montag auf Exkursion. Wir besuchten dabei das Interdisziplinäre Zentrum für Wissenschaftliches Rechnen (kurz IWR), genauer gesagt die Abteilung für Computergrafik, in Heidelberg. Wir fuhren mit der S-Bahn und kamen dank Sigmunds und Daniels Ortskenntnis rechtzeitig zu den Vorträgen an. Zuvor konnten wir noch ein foucaultsches Pendel, mit dem man die Erdrotation veranschaulichen kann, betrachten.

Das Foucaultsche Pendel

Doch zu den Vorträgen: Der erste der drei beschäftigte sich mit der sogenannten Poincaré-Vermutung, die sich mit Drei-Sphären und Dreimannigfaltigkeiten, sprich Objekten im vierdimensionalen Raum, beschäftigt – ein Vortrag aus dem Bereich der reinen Mathematik. Als zweites erklärte uns Hubert Mara, der das Programm in Heidelberg freundlicherweise für uns organisiert hatte, etwas über das 3D-Scannen. Hierbei können mit unterschiedlichen Techniken dreidimensionale Modelle eines Objekts erstellt werden, deren Genauigkeit je nach Größe des Objekts zwischen Bruchteilen von Mikrometern, zum Beispiel bei etwa 5000 Jahre alten Keilschrifttafeln, und etwa einem halben Zentimeter, zum Beispiel bei der Vermessung eines Gebäudes, liegt. Alles in allem ist die Technik sehr genau und wird vor allem von Archäologen und Geographen gerne genutzt. Als drittes und letztes wurde uns eine Einführung in die

Computerlinguistik gegeben und die prinzipielle Funktionsweise von Suchmaschinen, wie zum Beispiel Google oder Bing, erklärt. Hierzu hatten wir im Anschluss auch noch Gelegenheit, ein von Studenten geschriebenes, sehr vereinfachtes Suchmaschinen-Programm zu testen, doch dazu später mehr.

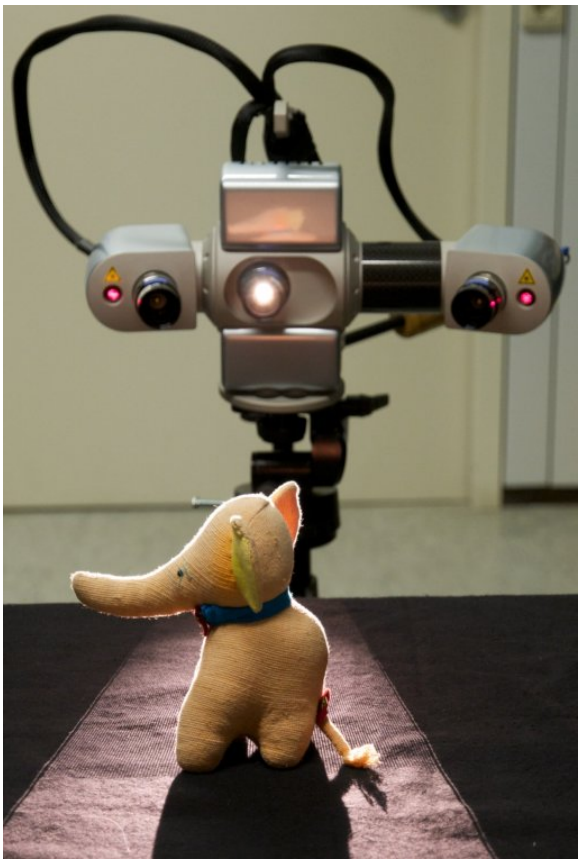
Nach den Vorträgen durften wir ein paar der dortigen Highlights anschauen und auch ausprobieren. Im Einzelnen waren das:

Ein 3D-Scanner mit dem wir ein räumliches Modell von Sigmund ermittelten. Dieser funktioniert nach dem Prinzip der Lichtstreifenprojektion. Dabei werden mit einem Projektor helle und dunkle Streifen auf das zu scannende Objekt projiziert. Dieses Muster wird dann von zwei Kameras, die in verschiedenen Winkeln zum Projektor ausgerichtet sind, aufgezeichnet. Von einer passenden Software wird nun anhand der Verzerrung des Musters auf der Objektoberfläche ein dreidimensionales Modell errechnet. Um ein vollständiges Modell zu erhalten, werden etwa acht bis zehn Scanvorgänge benötigt, da immer nur eine Seite des Objekts gescannt werden kann. Weil Sigmunds Ohren während des Scannens leider ein wenig wackelten, war das Modell am Ende leider ebenfalls verwackelt.

Auch durften wir ein Programm ausprobieren, das eine Landschaft mit zwei Tempeln darstellte, die man dann durch eine 3D-Brille räumlich wahrnehmen konnte. In ihr konnte man mithilfe einer Art Joystick durch Senken oder Heben seines Kopfes navigieren. Das war sehr beeindruckend und hat allen viel Spaß gemacht, zumal es sich bei den Tempeln um sehr detailgetreue Nachbildungen handelte.

Als letztes testeten wir das Suchmaschinen-Programm. Dieses kann anhand von Beschreibungen eines Bildes Suchbegriffe bestimmten Bildern zuordnen. Sprich, man konnte einen Suchbegriff oder einen Satz eingeben und das Programm fand heraus, welches der etwa 20 Bilder denn nun gemeint sein könnte. Auch das war sehr interessant, da wir ja im Kurs selbst schon viel programmiert hatten.

An dieser Stelle möchten wir uns ganz herzlich bei allen Mitarbeitern des IWR Heidelberg bedanken, die für uns dieses abwechslungsreiche und lehrreiche Programm auf die Beine gestellt haben.



Sigmund beim 3D-Scannen

Anekdoten aus der Kursarbeit

Sebastian Dallinger

Die eine oder andere Kurssitzung wurde durch spontane Kommentare und interessante Formulierungen deutlich aufgelockert. Um sie alle wiederzugeben, würde man eine Extradokumentation benötigen. Daher werden wir hier nur eine Auswahl zum Besten geben:

- **Alex:** „Ist a größer, wird das Objekt schneller schneller.“
  Lukas: „Und wenn a größer wird, aber noch negativ ist? “
  Alex: „Dann wird das Objekt eben weniger schnell langsamer.“
- **Alex:** „2006 hab ich mal die Theater-KüA geleitet!“
  Daniel: „Das war das Jahr, in dem ich beim Theater eingeschlafen bin.“

- **Alex:** „Wenn die Kugel sich nicht bewegt, tut es nicht weh, wenn man sie an den Kopf kriegt.“
- **Alex:** „Alle Körper fallen gleich schnell.“
  Lässt später Kreide und Papier fallen; die Kreide fällt schneller.
  Daniel: „Du hast gerade das widerlegt, was du vorhin an die Tafel geschrieben hast.“

Nachwort

Die Kursleiter

Das Ergebnis kann sich definitiv sehen lassen – doch noch viel wichtiger ist die Art und Weise, wie es erreicht wurde: von C++ und objektorientierter Programmierung, über Mathematik und dreidimensionale Grafiken zu Kinematik, Kollision und rollenden Kugeln – sämtliche das Billard betreffende Felder wurden abgedeckt. Aber nicht nur das Fachliche wird uns alle auf unserem weiteren Lebensweg begleiten: das Menschliche hatte immer einen festen Platz; durch Sportfest, Billardturnier, oder einfach das Billardspielen zwischendurch, festigten wir das Netz, das uns alle verbindet, ganz egal, ob Teilnehmer oder Kursleiter.