

Breakout, Flipper & Friends



Der Breakout-Kurs

Vorwort

In unserem Kurs „Breakout, Flipper & Friends“ beschäftigten wir uns mit verschiedenen Teilgebieten der Informatik, Physik und Mathematik. Das Ziel des Kurses war es, innerhalb der zwei Wochen in Adelsheim ein funktionstüchtiges Computerspiel zu programmieren.

Diese Aufgabe wurde mit viel Freude und Spaß bewältigt. In unserem Kurs herrschte dadurch eine gute, lockere Stimmung, die uns allen zwei unvergessliche Wochen bereitete.

In den folgenden Artikeln wollen wir einen

kleinen Einblick in unsere Arbeit während dem Eröffnungswochenende und der Akademie geben.



Unser Kurs stellt sich vor

... die Kursleiter

Daniel Jungblut

Geburtstag: 23.04.1984

Wohnort: Heidelberg

„Seh ich so aus, wie wenn ich wüsste was ich an die Tafel schreibe? Nein? Dann is gut!“

Daniel studiert Mathematik und Informatik an der Universität Heidelberg. Er leitete den Kurs zum dritten Mal und hat sehr viel Erfahrung im Programmieren. Er leitete zusammen mit Aaron die Mathe-KüA über Wahrscheinlichkeitsrechnung.

Thomas Zessin

Geburtstag: 28.05.1983

Wohnort: Ladenburg

„Ich bin älter als Daniel!“

Thomas studiert an der Universität Heidelberg Chemie und Mathematik. Während der zwei Wochen in Adelsheim musste er eine Klausur ablegen, die ihn einen Schritt weiter zu seinem Traumberuf „Lehrer“ brachte. Er verdeutlichte uns unter anderem die Vektorrechnung.

Alexander Schlüter

Geburtstag: 10.02.1989

Wohnort: Leutenbach

„Daniel, warum hast du eigentlich so eine lange Nase?“

Alex war unser 17jähriger Schülermentor, der 2004 selbst in diesem Kurs war. Alex hatte eine sehr gute innere Uhr, die sich immer pünktlich zum Mittagessen meldete. Da er sich sehr für Physik interessierte, erklärte er uns den physikalischen Teil unserer Kursarbeit. Außerdem leitete er die Physik-KüA über die Relativitätstheo-

rie sowie zusammen mit Sandra und Janne die Theater-KüA.



... die Kursteilnehmer

Karima Benimmar

Geburtstag: 17.02.1991

Wohnort: Sindelfingen

„Ich muss ja *mampf* meinem Ruf gerecht werden!!!“

Karima saß wie alle Mädchen in der zweiten Reihe, was angeblich typisch für Mädchen sei. Sie kümmerte sich gern um das Design unseres Spieles, wobei es bei der Farbwahl der einzelnen Bestandteile öfters Meinungsverschiedenheiten gab. Auch sie sorgte dafür, dass wir stets pünktlich zum Mittagessen kamen. Außerdem spielte sie im Orchester Klarinette und besuchte die Poker-KüA.

Claudia Denninger

Geburtstag: 19.09.1991

Wohnort: Stuttgart

„Wir müssen uns noch Zitate ausdenken...“

„Mama“ von Ella. Claudia designte zusammen mit Ella einige Levels und beschäftigte sich mit der Erstellung des Menüs. Sie spielte Klarinette im Orchester und besuchte regelmäßig die Tanz-KüA.

Johannes Henrichsmeyer

Geburtstag: 07.10.1991

Wohnort: Rosenfeld

„*Gähn* ... reicht des???“

Johannes und Benni haben wir die Goodies in unserem Spiel zu verdanken. Seine Kamera hatte er fast immer dabei und sorgte für viele lustige Fotos, u. a. von unseren Kursleitern. Außerdem rockte er in der Ska-Band an der Gitarre und war Mitgründer der außergewöhnlichsten KüAs.

Lin Jin

Geburtstag: 07.07.1989

Wohnort: Karlsruhe

„Was soll ich sagen???“

Auch Lin saß in der typischen Mädchen-Reihe und arbeitete mit den anderen drei Mädels am Design unseres Spieles. Bei der Rotations-Vortrags-Probe meinte Daniel sie würde den UpM (Unds pro Minute)-Rekord knacken. Dadurch, dass Lin Chinesisch konnte, fiel ihr die Kommunikation mit Alex und Vicky besonders leicht.

Aaron Kimmig

Geburtstag: 18.07.1993

Wohnort: Aachern

„Ich hab noch keine *mampf*, *würg*, *schluck*, Apfelringe!!!“

Aaron arbeitete mit Thomas an der Vektor-kollision und an unserem drehbaren Paddle. Als mathebegeisterter Teilnehmer leitete er zusammen mit Daniel die Mathe-KüA über Wahrscheinlichkeitsrechnung.

Jonathan Krüger

Geburtstag: 26.05.1991

Wohnort: Riederich

„Ich bin gar nicht müde... *zZzZzZz*“

Jonathan musste uns leider krankheitsbedingt früher verlassen. Nichts desto trotz

half er tatkräftig beim Programmieren des Spiels mit. Die Idee, Flying Objects in unser Spiel einzubauen, kam von ihm, außerdem übersetzte er für Alex und Vicky den Kursinhalt sehr geduldig ins Englische. Das Theaterspielen machte ihm sehr viel Spaß, außerdem spielte er Trompete im Orchester.

Nikolas Kuhn

Geburtstag: 13.04.1992

Wohnort: Rheinfelden

„Des passt hier so schön rein...“

Nikolas war unser Allrounder. Von Vektor-kollision bis hin zu Levels programmierte er einfach alles, was gebraucht wurde. Er war begeistert von allen Physik- und Mathe-KüAs und war deswegen immer dabei.

Benjamin Maier

Geburtstag: 19.02.1991

Wohnort: Waiblingen

„Ein Breakout ohne Goodies ist wie ein Programmierer ohne Hunger!!“

Unser Benni beschäftigte sich am liebsten mit dem Erfinden von Goodies. Seine Idee war es ein Gravitationsgoodie zu programmieren, das den Ball immer wieder nach unten zieht.

Leonhard Markert

Geburtstag: 18.04.1992

Wohnort: Gaggenau

„Die Baaaaaaaaaaaaaaaaa..z-Kraft“

Unser modebewusster Durchchecker beschäftigte sich am liebsten mit den Problemen die die anderen nicht hinbekamen. Er hatte immer gute Ratschläge auf Lager, die uns sehr weiterhalfen. Außerdem spielte er mit seiner Posaune im Orchester und besuchte die Tanz-KüA.

Ella Schmidt

Geburtstag: 22.01.1992

Wohnort: Kornwestheim

„Ja Claudia, is gut Claudia“

Auch Ella saß in der Mädchenreihe und arbeitete kräftig am Design mit. Ihr und Claudia haben wir die tollen Ideen für neue, interessante Levels, insbesondere das letzte Level und die schönen Bilder unseres Spiels zu verdanken. Ella besuchte neben den Physik- und Mathe-KüAs auch die Poker-KüA.

Vicky

Vicky war eine unserer chinesischen Gäste. Sie half uns viel bei der Vorbereitung zur Rotation und zur Abschlusspräsentation. Während der zwei Wochen in den Sommerferien beschäftigte sie sich mit dem designen verschiedener Levels.

Alex

Alex war der zweite Chinese unseres Kurses. Er hat die meisten Levels designt. Außerdem war er bei der nachmittäglichen Sport-KüA immer begeistert dabei. Durch Johannes Hilfe lernte er auch schnell die wichtigsten deutschen Sätze, wie „Ich habe Hunger“ und „Du bist schön.“. Außerdem war er Mitglied der Chinese-Boy-Group.



Das Eröffnungswochenende und die Zeit vor der Akademie

CLAUDIA DENNINGER, ELLA SCHMIDT

Wie schon erwähnt war das Ziel unseres Kurses ein Computerspiel zu programmieren, genauer gesagt, ein Breakout-Spiel.

Um programmieren zu können mussten wir zuerst lernen mit der Programmiersprache C++ und der dazugehörigen Entwicklungsumgebung Dev C++ umzugehen. Da nur sehr wenige von uns Programmiererfahrung hatten, lernten wir als erstes die Grundlagen.

Wir fingen dann auch Schritt für Schritt mit einfachen Programmen an, die z. B. Zahlen ausgeben und berechnen konnten. Außerdem lernten wir, wie man mit Hilfe der Grafik-Bibliothek OpenGL Kreise, Vielecke und Linien zeichnen kann.

Danach war das Eröffnungswochenende auch schon wieder viel zu schnell vorbei. Damit wir unsere Programmierkenntnisse weiter ausbauen konnten, bekamen wir alle zwei Wochen von unseren Leitern einen Übungszettel mit Aufgaben per e-Mail. Wenn die Aufgaben zu Problemen geführt haben, diskutierten wir im Forum darüber und gaben uns gegenseitig Tipps. Als dann schließlich wieder alle guter Laune und gespannt auf die kommenden zwei Wochen in Adelsheim anreisen konnten wir einen Ball zwischen vier Banden hin und her fliegen lassen ...

Grundlagen

JONATHAN KRÜGER, AARON KIMMIG

Wie denkt ein Computer?

Die Computersprache ist für uns sehr schwer zu verstehen, denn sie besteht nur aus Nullen und Einsen. Anders gesagt: Der Computer kennt nur „Strom ein“ und

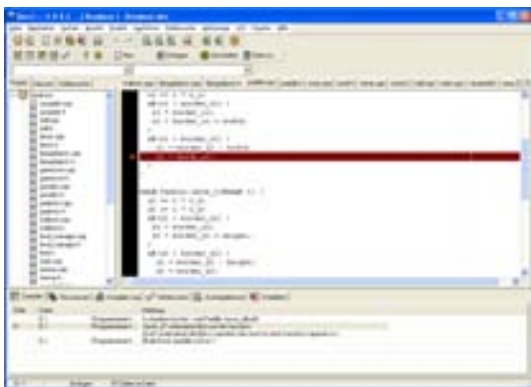
„Strom aus“. Wir nennen dieses System „binäres System“.

Um dem Computer Befehle zu geben, müssen wir ihm also entweder Befehle in binärer Form geben oder eine Programmiersprache verwenden, die wir verstehen können. Für unser Breakout-Spiel verwenden wir die Programmiersprache C++. Damit der Computer diese Sprache verstehen kann gibt es den Compiler, der den Programmcode in Maschinsprache übersetzt.

Wir verwendeten zum Programmieren die Entwicklungsumgebung „Dev-C++“. Dieses Programm bietet nützliche Hilfsmittel wie z. B. Fehlererkennung, Buttons und Dateiansichten.

Aufbau der Entwicklungsumgebung

In der Entwicklungsumgebung Dev-C++ können wir in einem Fenster, dem sogenannten Editor, den Programmcode eingeben.



Was braucht man zum Programmieren?

Befehle:

Das Vokabular der Programmiersprache sind Befehle. Befehle reichen von der Addition von Zahlen bis zum Darstellen einer sich bewegenden Kugel auf dem Bildschirm. Im Folgenden werden einzelne wichtige Befehlsklassen erklärt.

Variablen:

Sehr wichtig ist beim Programmieren auch der Umgang mit Variablen. Diese bieten die Möglichkeit Werte zu speichern und zur Laufzeit Werte zu verändern. Jede Variable gehört dabei einem Typ an (z. B. ganze Zahl, Gleitkommazahl, Text).

Verzweigung:

Durch eine Verzweigung wird der Computer dazu veranlasst eine Entscheidung zu treffen. Dies ist z. B. wichtig, um zu erkennen, ob der Ball in unserem Spiel mit einem Stein kollidiert oder nicht.

Schleifen:

Eine weitere wichtige Vereinfachung beim Programmieren stellen die Schleifen dar. Wenn beispielsweise der Computer die Zahlen von 1 bis 100 schreiben soll, kann man entweder 100 Schreibbefehle eingeben oder eine Schleife verwenden, in der dieser Schreibbefehl nur einmal vorkommt. Für die Zahl wird eine Variable eingesetzt, die nacheinander den Wert von 1 bis 100 annimmt. Wenn die Variable den Wert 100 erreicht hat, wird die Schleife beendet. Auch ist manchmal beim Programmstart noch nicht klar, wie oft eine Aktion durchgeführt werden muss. In diesem Fall wäre es nicht möglich ohne Schleifen auszukommen.

Funktionen:

Funktionen sind nützlich, wenn wir bestimmte Befehlsabfolgen, meistens komplizierte Berechnungen, an verschiedenen Stellen im Programm mehrmals benötigen. Diese Befehlsabfolgen können wir in eine Funktion schreiben. Nun können wir an jeder Stelle im Programm diese Funktion aufrufen. Das Programm wird auf diese Weise übersichtlicher.

Grafikfenster:

Wenn man nun aber etwas zeichnen will, benötigt man ein Fenster, indem die vom Computer errechneten Bilder angezeigt werden. In diesem Fenster orientiert man

sich in einem Koordinatensystem. Vor allem Grafiken, aber auch Texte können in diesem Fenster ausgegeben werden. Man muss nur die Koordinaten angeben, wo der Text platziert wird.



Farben:

Ein Spiel würde langweilig aussehen, wenn es nicht bunt wäre. Die Grafikkbibliothek OpenGL bietet die Möglichkeit, Farben nach der „additiven Farbmischung“ zu erstellen. Das bedeutet, dass wir alle Farben mit rot, grün und blau mischen können, dabei können wir auch das Verhältnis der Farben zueinander und die Farbintensität ändern. Wenn wir alle Farben mit voller Intensität zusammenmischen, ergibt dies die Farbe weiß.

Texturen:

Texturen sind fertige Bilder, die wie Tapeten auf Objekte im Grafikfenster geklebt werden können und so das Spiel schöner machen. Beispielsweise ein Goodie: in unserem Spiel ein Stein, auf dem eine Textur platziert wurde.



Im Beispielbild ist die Textur für das Goodie eine Rakete. Dies ist gleichbedeutend mit dem Zweck des Goodies, denn es macht den Ball schneller. Durch Texturen werden die Goodies also interessanter und ihre Funktion verständlicher.

Farben in C++

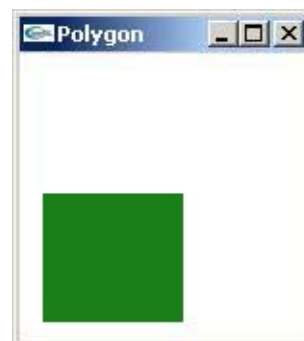
CLAUDIA DENNINGER, ELLA SCHMIDT

Das RGB-System

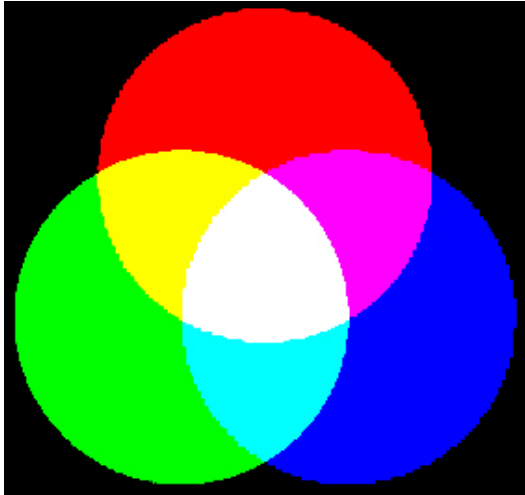
Der Computer arbeitet mit dem RGB (rot-grün-blau) Farbsystem und der additiven Farbmischung. Mit dem Befehl „glColor3f“ geben wir im Quelltext für jede dieser drei Farben ein Wert zwischen 0 und 1 an:

```
glColor3f(0.1, 0.5, 0.1);
```

Die erste Zahl steht für die Farbintensität von rot, die zweite Zahl für grün und die dritte für blau. Hat die Farbe den Wert 0, wird die Farbe nicht verwendet, bei 1 wird die Farbe so hell wie möglich verwendet. Die Farben werden mit ihren verschiedenen Intensitäten gemischt, so entsteht schließlich eine Mischfarbe (in diesem Fall dunkelgrün).

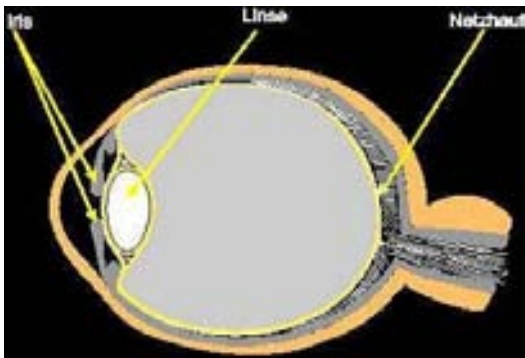


Mit diesem System ist es uns möglich fast alle Farben (16 Millionen) darzustellen. Die folgende Grafik verdeutlicht die Mischung der Grundfarben.



Warum benutzen wir das RGB-System?

Auf der Netzhaut unseres Auges befinden sich drei verschiedene Sorten von Farbrezeptoren, die ebenfalls auf rot, grün und blau reagieren:



In unserem Gehirn entstehen dann durch Kombination der Einzelfarben verschiedene Mischfarben.

Bewegung

LEONHARD MARKERT

Definition (Frame). Ein Frame ist ein Bild von vielen, die vom Computer in schneller Abfolge berechnet werden. Durch

die Aneinanderreihung vieler Frames entsteht der Eindruck von Bewegung.

Umsetzung im Programm

Die Grundlage für die Bewegung von Objekten in unserem Spiel ist folgende Formel:

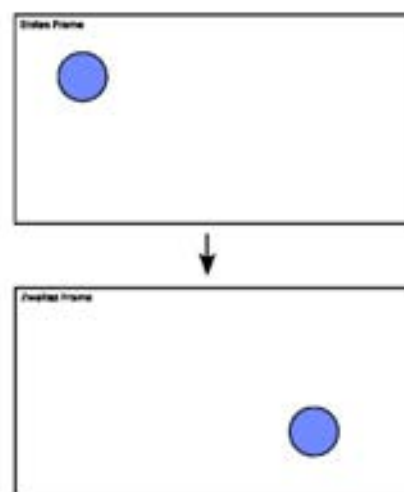
$$\Delta s = v \cdot \Delta t$$

Konkret bedeutet das für den Ball, dass seine neue Position wie folgt berechnet wird:

$$x_{neu} = x_{alt} + v_x \cdot \Delta t$$

$$y_{neu} = y_{alt} + v_y \cdot \Delta t$$

Hierbei steht x_{neu} für die neue, zu berechnende x-Position, x_{alt} für die x-Position im vorherigen Frame, v_x für die Geschwindigkeit in x-Richtung und Δt für die seit dem letzten Frame vergangene Zeit steht. Analog gilt dies für y_{neu} , y_{alt} und v_y .



Frames

Diese Art der Bewegung funktioniert auch für die Flying Objects und für die Goodies. Die Goodies bewegen sich allerdings nur in negative y-Richtung (nach unten).

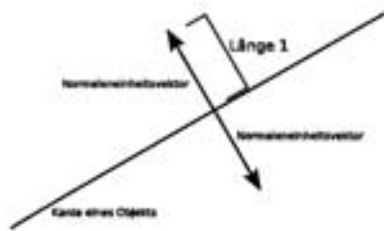
Kollision und Reflexion

LEONHARD MARKERT

Kollision

Um herauszufinden, ob eine Kollision zwischen der Kugel und einem anderen Objekt stattfand, wird jeweils vor der Berechnung eines neuen Frames für jeden Stein, die Banden, die Flying Objects und das Paddle geprüft, ob der Ball mit einem dieser Objekte kollidiert ist.

Zur Kollisionsabfrage benutzen wir Normalenvektoren. Ein Normalenvektor ist ein Vektor, der senkrecht auf einer Gerade oder Strecke steht. Im zweidimensionalen Raum hat jede Strecke zwei Normaleneinheitsvektoren (Normalenvektoren mit Betrag 1).



Normaleneinheitsvektoren

Auf unser Spiel bezogen heißt das, dass alle Kanten der Banden, der Flying Objects oder des Paddles je zwei Normaleneinheitsvektoren haben.

Für die Reflexion entscheidend sind die Normaleneinheitsvektoren, die „nach außen“ zeigen. Mithilfe dieser Normaleneinheitsvektoren errechnet man den gerichteten Abstand $d(\circ$: Skalarprodukt):

$$(x - p) \circ n$$

mit x Ballmitte, p beliebiger Kantenpunkt, n Normaleneinheitsvektor. Ist nun dieser gerichtete Abstand für alle Kanten negativ, so befindet sich die Kugel in dem Objekt. Also hat eine Kollision stattgefunden.

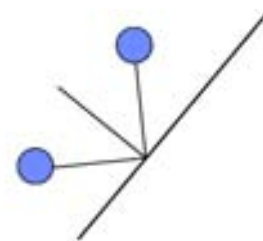
Reflexion

Wurde eine Kollision registriert, wird die Reflexion eingeleitet. Diese besteht aus zwei Schritten:

1. Heraussetzen der Kugel aus dem Objekt mit welchem sie kollidiert ist,
2. Berechnen des neuen Geschwindigkeitsvektors v_{neu} :

$$v_{neu} = v_{alt} + 2n \cdot (-v_{alt} \circ n)$$

(n Normaleneinheitsvektor der Kante)



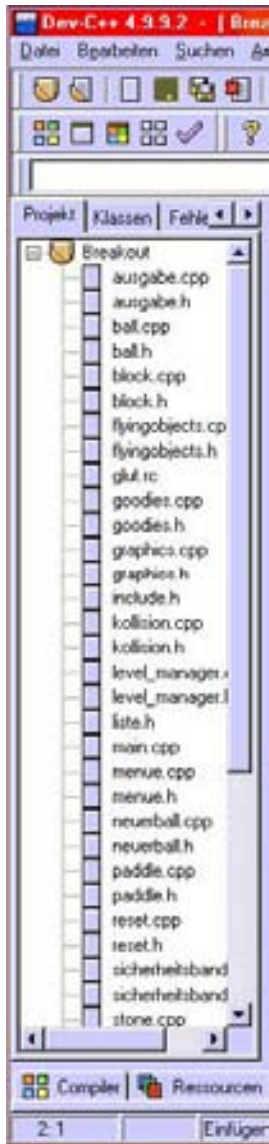
Reflexion

Objektorientiertes Programmieren

JOHANNES HENRICHSMEYER, KARIMA BENIMMAR



Unsere ersten Programme bestanden nur aus einer Datei. Im Laufe der Zeit wurde der Quelltext immer länger und damit unübersichtlicher.



Klassenansicht des
Breakout-Programms

Um die verschiedenen Bestandteile unseres Spiels (Paddle, Bande, Ball, u. s. w.) leichter programmieren zu können, lernten wir die Methode des Objektorientierten Programmierens. Damit ist es möglich sehr nahe am menschlichen Denken zu programmieren, weil Objekte beim Programmieren so beschrieben werden, wie wir sie wahrnehmen: „Wie sieht das Objekt aus?“ (Unser Ball würde hier z. B. beschrieben durch Farbe, Position und den Radius) und „Was kann das Objekt?“ (Der Ball kann gezeichnet werden und sich bewegen).

Das Objektorientierte Programmieren ist

eine Arbeitserleichterung für den Programmierer.

Wie funktioniert Objektorientiertes Programmieren?

Gegenstände (wie die Bande) oder Aktionen (wie die Kollision) in unserem Spiel sollen sich unserer Wahrnehmung entsprechend beschreiben und programmieren lassen.

Das geschieht, indem sogenannte Klassen für die Objekte angelegt werden. Klassen sind Baupläne, die beschreiben, wie ein Objekt aufgebaut ist (ähnlich wie der Konstruktionsplan eines Autos).

In den Klassen wird das Objekt durch seine Eigenschaften und seine Methoden beschrieben.

Eigenschaften sind Informationen, die das Objekt beschreiben. Die Eigenschaften des Paddles sind beispielsweise seine Position, Geschwindigkeit, Farbe, Breite und Höhe. Für die Werte der Eigenschaften wird Speicherplatz reserviert.

Die Methoden beschreiben durch Funktionen alle möglichen Handlungen eines Objekts. Eine Methode des Paddles ist zum Beispiel die Bewegung.

Mit Hilfe der Baupläne können in der Hauptdatei dann beliebig viele Objekte einer Klasse erstellt werden. Den Eigenschaften werden Werte zugewiesen: Das Paddle bekommt seine Anfangsposition, sein Aussehen und seine Geschwindigkeit. Diesen Zuweisungsvorgang nennt man „Initialisierung“:

```
paddle.init(165, 10, 235, 25,
0.9, 0.5, 0.0);
```

Die verschiedenen Zahlen stehen für die Positions- und Farbwerte.

Wir legten für jede Klasse eine neue Datei an, was das parallele Arbeiten an unserem Projekt vereinfachte. Jede Gruppe spezialisierte sich auf ein Element des Spiels,

so gab es zum Beispiel eine „Goodiegruppe“ oder eine Gruppe, die für das Leveldesign zuständig war. Die Gruppen legten ihre Klassen an, arbeiten an ihnen und mussten nur wenige Zeilen in die Hauptdatei einfügen. So konnten die Dateien später leicht zu einem Programm zusammengefügt werden.

Vererbung

Viele Objekte besitzen ähnliche Eigenschaften. Die Bande und das Paddle sind beispielsweise beide rechteckig und können den Ball reflektieren. Es ist möglich, zunächst nur einen Bauplan für die Bande zu erstellen und ihn dann für das Paddle zu erweitern, dass nennt man Vererbung.

Zusammenfassung der Vorteile des OOP

- Objektorientiertes Programmieren ist an das menschlichen Denken angelehnt
- Es entsteht eine Aufteilung des Programms in verschiedene Dateien, wodurch das Programm strukturierter und übersichtlicher wird
- Die Hauptdatei wird kürzer, da weniger Befehlszeilen nötig sind, um ein Objekt zu erstellen
- Es muss nicht für jedes Objekt der komplette Code neu geschrieben werden
- Eine Funktion kann für mehrere Objekte verwendet werden, wie zum Beispiel die Kollision des Balls mit Bande, Stein oder Paddle

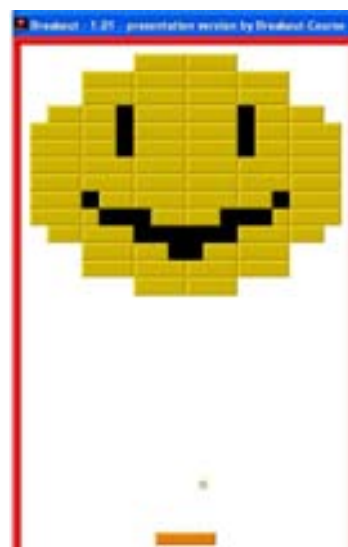
Unser Spiel

LIN JIN

Was ist ein Breakoutspiel ?

Breakout ist ein Computerspiel, dessen Ziel es ist, mit einem Ball alle Steine abzuschie-

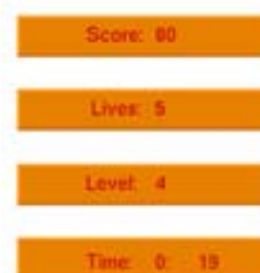
ßen, um dadurch in immer höhere Stufen (*Level*) zu gelangen und nach Möglichkeit alle Level zu meistern. Mit Hilfe der Tastatur ist es dem Spieler möglich ein Paddle zu bewegen um dadurch den Ball zu steuern. Verläßt der Ball den unteren Spielfeldrand, verliert man ein Leben. Ein bisschen Geschick gehört somit auch zum erfolgreichen Spielen von Breakout dazu.



Unser Spiel

Die ersten Überlegungen

Es war natürlich nicht unser Ziel, ein einfaches, langweiliges Spiel zu programmieren, sondern es entstanden viele kreative Ideen, die wir auch meistens gut umgesetzt haben. Verschiedene Levels und das Sammeln von Punkten sind wichtige Faktoren in einem guten Computerspiel.



Spielanzeige

All dies fügten wir in unser Spiel ein, zusätzlich gelang es uns eine Uhr, ein Menü sowie eine Hilfefunktion einzubauen und fliegende Hindernisse, die nach ihrem Erschaffer im Folgenden als Flying Objects bezeichnet werden.



Menü

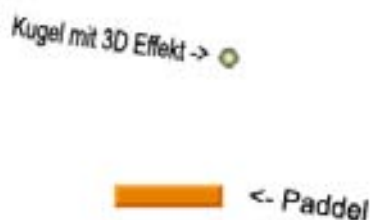
Mit Hilfe von Bildern, die wir als Texturen in das Spiel eingebaut haben, konnten wir auch erfahrenere Computerspieler begeistern. Zu diesen Bildern gehören zum Beispiel unser Kurslogo, der Gameover- und der Winnerbildschirm.



Spiel-Ende-Bildschirme

Komponenten des Breakout-Spiels

- Kugel, oder Ball:



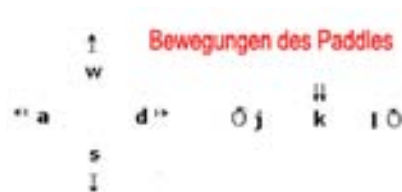
Mit ihr ist es möglich, die einzelnen Steine abzuschießen und somit in den Levels aufzusteigen. Da die Kugel keine Wände durchdringen kann, wird sie an ihnen reflektiert.

- Rahmen (Banden):

Der Rahmen ist die Begrenzung an den Seiten und oberhalb des Spielfelds.

- Paddle:

Das Paddle ist ein bewegliches Rechteck, das man mit Hilfe der Tastatur in verschiedene Richtungen verschieben und drehen kann. Mit dem Paddle muss der Ball abgefangen werden, zudem kann die Bewegungsrichtung des Balls mit Hilfe des Paddles geändert werden.



- Steine:



Steine sind rechteckige Blöcke, die man mit Hilfe der Kugel zerstören kann.

Manche Steine besitzen mehr als ein „Leben“, d. h. dass diese nach dem ersten Zusammenstoß mit der Kugel nicht sofort zerstört werden, sondern erst nach mehreren.

Stößt die Kugel mit einem Stein zusammen wird sie von ihm physikalisch korrekt reflektiert. Nach der Zerstörung eines Steins erhält man 10 Punkte. Wurden alle Steine abgeschossen, kommt man ins nächste Level.

- Flying Objects:

Die sogenannten Flying Objects sind fliegende Blöcke, die dazu da sind, das Zerstören der Steine zu erschweren. Sie bewegen sich frei in Richtungen, die der Spieler nicht beeinflussen kann und reflektieren die Kugel ebenfalls physikalisch korrekt. Flying Objects können allerdings nicht zerstört werden und dienen lediglich als Hindernisse.



- Level-, Punkte- und Lebenszähler sowie Uhr:

Rechts vom Spielfeld befinden sich einzelne Kästen, in denen sich Zähler befinden. Die Uhr wird bei Betätigen der Pausefunktion angehalten.

- Goodies:

Goodies sind Bonusobjekte des Spiels, die nach Zerstörung eines Steins nach unten fallen und mit dem Paddle eingesammelt werden können. Hierdurch werden bestimmte Effekte hervorgerufen.

Breiter-Goodie	
Schmaler-Goodie	
Schneller-Goodie	
Langsamer-Goodie	
Lebensgoodie	
Punkt-Goodie	
Goodie für Sicherheitsbande	
Durchschlagkraftsgoodie	
Schwerkraftsgoodie	

Eine Liste aller Goodies aus unserem Spiel

Physik in unserem Spiel

BENNI MAIER

Um in unser Programm Schwerkraft einbauen zu können, mussten wir zunächst über fallende Körper und deren Geschwindigkeitsverlauf Bescheid wissen. So beschäftigten wir uns mit Kinematik, der Lehre der Bewegung.

Gleichförmige Bewegung

Als erstes betrachteten wir eine gleichförmige Bewegung, also eine Bewegung mit konstanter Geschwindigkeit. Wir stellten die Formel $v = \frac{\Delta s}{\Delta t}$ auf, die wir schon aus der Schule kannten. Möchte man beispielsweise die zurückgelegte Strecke eines Autos wissen, das 11 Minuten lang durchschnittlich 100 km/h schnell fährt, so formt man diese Formel um zu:

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

Mit den Angaben kann man nun berechnen:

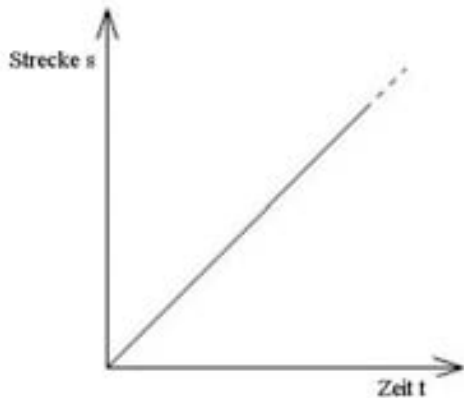
$$\Delta t = 11 \text{ min} = 660 \text{ s}$$

$$v = 100 \text{ km/h} = 27,8 \text{ m/s}$$

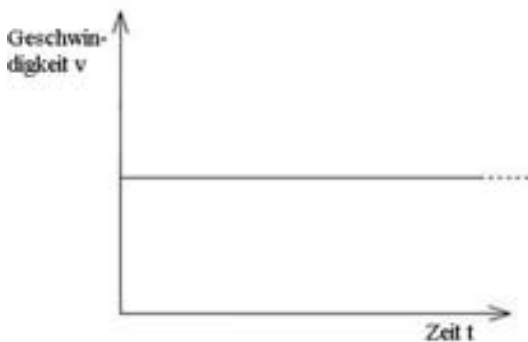
$$\Rightarrow \Delta s = v \cdot \Delta t = 27,8 \text{ m/s} \cdot 660 \text{ s}$$

$$\approx 18333 \text{ m} \approx 18,3 \text{ km}$$

Betrachten wir ein Strecke-Zeit-Diagramm (s - t -Diagramm), erhalten wir eine steigende Ursprungsgerade. Im Geschwindigkeit-Zeit-Diagramm (v - t -Diagramm) verläuft die Gerade waagrecht, da die Geschwindigkeit über die ganze Zeit konstant ist.



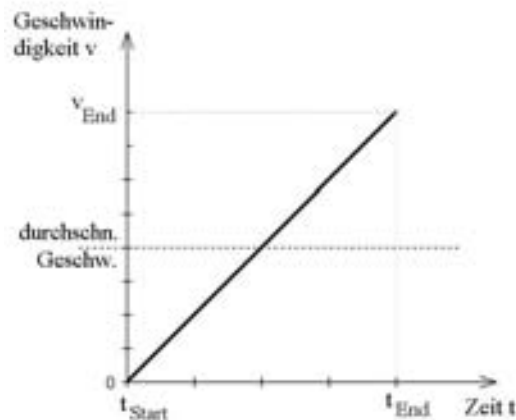
s - t -Diagramm



v - t -Diagramm

Beschleunigte Bewegung

Man betrachtet nun eine gleichmäßig beschleunigte Bewegung. Hier soll mit fortschreitender Zeit die Geschwindigkeit zunehmen. Die Gerade im v - t -Diagramm steigt an.



Gleichmäßig beschleunigte Bewegung:
 v - t -Diagramm

Die Steigung der Gerade nennt sich die Beschleunigung a . Festgelegt ist sie als Geschwindigkeitsänderung Δv pro Zeitspanne Δt

$$\Rightarrow a = \frac{\Delta v}{\Delta t}$$

Ihre Einheit ist

$$1 \frac{m/s}{s} = 1m/s^2$$

Wenn nun in unserem Beispiel das Auto anfahren soll, also aus dem Stand beschleunigt, können wir die Formel in dieser Form verwenden:

$$a = v_{end}/t_{end}$$

Hierbei ist die Geschwindigkeitsänderung Δv gleich der Endgeschwindigkeit v_{end} , da der Startwert $v_{start} = 0$ ist und somit gilt:

$$\Delta v = v_{end} - v_{start} = v_{end} - 0 = v_{end}$$

Das gleiche gilt für Δt .

In einem solchen Beispiel mit konstanter Beschleunigung beträgt die Durchschnittsgeschwindigkeit \bar{v}_d die Hälfte der Endgeschwindigkeit v_{end} :

$$\bar{v}_d = \frac{1}{2} \cdot v_{end}$$

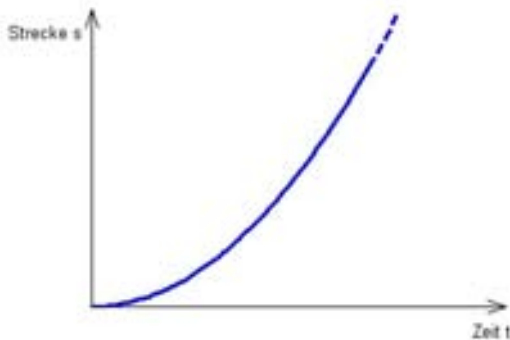
$$\Rightarrow \Delta s = \frac{1}{2} \cdot v_{end} \cdot t_{end}$$

Die schon erwähnte Formel $a = \frac{v_{end}}{t_{end}}$ können wir zu $v_{end} = a \cdot t_{end}$ umformen und in die obige Formel einsetzen:

$$\Delta s = \frac{1}{2} \cdot (a \cdot t_{end}) \cdot t_{end}$$

$$\Delta s = \frac{1}{2} \cdot a \cdot t_{end}^2$$

Somit erhalten wir eine wichtige Formel zur Berechnung von Beschleunigungen aus dem Stand. Im s - t -Diagramm dargestellt ergibt diese Funktion eine Parabel:



Gleichmäßig beschleunigte Bewegung:
 s - t -Diagramm

Beispiel: Ein Auto mit (gleichmäßiger) Beschleunigung $0 - 100 \text{ km/h}$ in 8 Sekunden fährt los. Welche Strecke hat es nach 11 s zurückgelegt?

Beschleunigung a : $100 \text{ km/h} = 27,8 \text{ m/s}$

$$\Rightarrow a = \frac{v_{end}}{t_{end}} = \frac{27,8 \text{ m/s}}{8 \text{ s}} \approx 3,47 \text{ m/s}^2$$

Strecke s :

$$\Delta s = \frac{1}{2} \cdot a \cdot t_{end}^2$$

$$\Delta s = \frac{1}{2} \cdot 3,47 \text{ m/s}^2 \cdot (11 \text{ s})^2 \approx 210,07 \text{ m}$$

Beschleunigung und Kraft

Wir verifizierten unsere Formeln durch Versuche auf einer Luftkissenfahrbahn. Dabei erkannten wir, dass die Beschleunigung eines Körpers von der auf ihn wirkenden Kraft abhängt und dazu proportional ist:

$$a \sim F$$

Außerdem hängt die Kraft auch von der Masse des beschleunigten Körpers ab:

$$F \sim m$$

Daraus folgt:

$$F \sim m \cdot a$$

bzw.

$$F = \alpha \cdot (m \cdot a)$$

wobei α ein konstanter Wert ist. Die Einheit der Kraft Newton ist so definiert, dass $\alpha = 1$ gilt. Somit gilt:

$$F = m \cdot a$$

Um ein Gewichtsstück der Masse 1 kg mit $a = 1 \text{ m/s}^2$ zu beschleunigen, wird also eine Kraft von genau 1 N benötigt:

$$1 \text{ N} = 1 \text{ kg} \cdot 1 \text{ m/s}^2$$

Dieses sog. 2. Newton'sche Axiom wird als Grundgleichung der Mechanik bezeichnet.

Der freie Fall

Für die Gravitation in unserem Programm mussten wir uns mit dem freien Fall beschäftigen. Hierbei spielt die Erdanziehungskraft eine wichtige Rolle. Sie wird mit dem Ortsfaktor g angegeben. Auf der Erde wirkt auf ein Gewichtsstück der Masse 1 kg eine Kraft von etwa $9,81 \text{ N}$, was sich experimentell herausfinden lässt. Anders ausgedrückt: Der Ortsfaktor g_{erde} beträgt

$$9,81 \frac{N}{kg} = 9,81 \frac{kg \cdot m/s^2}{kg} = 9,81 \frac{m}{s^2}$$

Nun haben wir die Einheit der Beschleunigung. Beim freien Fall gilt also $g = a$ und entsprechend

$$\Delta v = g \cdot \Delta t$$

Ausgehend von unseren Experimenten und den dazugehörigen theoretischen Überlegungen, fügten wir in unser Programm folgende Formel für die Gravitationskraft ein:

$$v_{y_{neu}} = v_{y_{alt}} - 300 \cdot \Delta t$$

Hierbei ist $v_{y_{neu}}$ die neue Geschwindigkeit des Balls in vertikaler Richtung, $v_{y_{alt}}$ die alte Geschwindigkeit des Balls in vertikaler Richtung und Δt die Zeit, die seit dem letzten berechneten Frame vergangen ist. Als Ortsfaktor haben wir den Wert 300 gewählt, da unser Programm ohne Einheiten arbeitet und dieser Wert eine schöne Flugkurve erzeugt.

Die Geschwindigkeit des Balls in horizontaler Richtung ist unabhängig von der Schwerkraft und wird in unserem Programm weiterhin so berechnet wie bisher.



Rotation und Abschlusspräsentation

NIKOLAS KUHN

Rotation

Am Freitag, den 8. September fand die Rotation statt. Dabei wurden alle Kurse in vier Gruppen (A, B, C, D) aufgeteilt. Die jeweils gleichnamigen Gruppen stellten sich untereinander die bisherigen Ergebnisse ihrer Kurse und den in den Kursen gelernten Inhalt vor. So bekam man einerseits einen Einblick in die anderen Kurse und konnte andererseits für die Abschlusspräsentation üben. Für unsere Präsentation benutzten wir Powerpoint. Den Vortrag hielten wir auf Deutsch, benutzten dabei aber englische Folien, damit auch die chinesischen Schüler folgen konnten. Unsere Präsentation beinhaltete folgende Themen:

- Was ist ein Breakoutspiel?
Anhand eines Screenshots stellten wir die Bestandteile unseres Spiels vor
- Grundlagen
Eine kurze Einführung ins Programmieren mit C++ und OpenGL
- Bewegung
Wie wird die Bewegung auf dem Computer dargestellt?
- Kollision
Wie funktioniert die Kollision?
- Objektorientiertes Programmieren
Was ist das?
- Ausblick
Was wir an unserem Spiel bis zum Ende der Akademie noch verbessern wollten.
- Statistik
Einige interessante Fakten zu unserem Spiel. (Länge des Quelltextes, Entwicklungszeit. . .)
- Unser Spiel *Die Zuhörer durften eine Probeversion unseres Spiels spielen*

Abschlusspräsentation

Bei der Abschlusspräsentation am Mittwoch den 13. September stellten wir unsere Kursarbeit den Familien der Akademieteilnehmer vor. Wir benutzten hierfür weitgehend die gleiche Präsentation wie bei der Rotation, gestalteten unsere Vorträge jedoch weitgehend ausführlicher. Die Vorträge fanden ab 15:00 Uhr alle 45 Minuten statt. Wir bildeten zwei Gruppen, die sich abwechselten. Am Ende jedes Vortrags boten wir die aktuelle Version des Spiels zum Testen an. Zusätzlich gestalteten wir eine Stellwand mit Bildern und Informationen zu unserem Spiel und unserer Kursarbeit.



Unsere „Breakwand“

Nachwort

CLAUDIA DENNINGER, ELLA SCHMIDT,
BENNI MAIER

Wir denken alle gerne an die zwei Wochen im Sommer zurück, die wir in Adelsheim verbringen durften. Deshalb finden wir es sehr schade, dass sich unsere Wege schon wieder trennen müssen, doch hoffen wir, dass unsere Freundschaften trotzdem bestehen bleiben. An dieser Stelle wollen wir uns alle ganz herzlich bei unseren Kursleitern und auch bei unserem Schülermentor für die schöne Zeit bedanken.