PHYSICSSITE

http://www.physicssite.ch.vu/

Eine Webseite zum Erstellen von zweidimensionalen Physik-Simulationen

Maturaarbeit

Peter Güttinger
Kantonsschule Schaffhausen, Dezember 2010

Betreuer: Rainer Steiger

Inhaltsverzeichnis

Code auf Titelblatt: Auszug aus NumberTextField.java

Vorwort

Für mich war schon früh klar, dass ich eine Maturaarbeit mit Schwerpunkt auf Programmieren machen möchte. Nur die Wahl des Themas fiel mir nicht leicht, da die Themenvielfalt in der Informatik gigantisch ist. Auch musste ich mich für eine Programmiersprache entscheiden, was mir allerdings leicht fiel, da ich die Programmiersprache Java lernen wollte, welche Plattform-unabhängig ist, als Applet in eine Website eingebunden werden kann und keine kostenpflichtigen Programme erfordert.

Zuerst habe ich an ein dreidimensionales Rennspiel gedacht. Ich habe diese Idee allerdings bald wieder aufgegeben, da 3D-Modellierung ziemlich zeitaufwendig ist, wenn man sich nicht damit auskennt. Nach weiteren Überlegungen kam ich verblüffend rasch auf ein Thema, welches mir gefiel. Ich war damals mehr oder weniger aktiv auf einer Website, wo man ganz einfach per Drag&Drop zweidimensionale Spiele zusammenbauen kann[1]. Da die Entwickler dort inaktiv waren und bestehende Bugs nicht behoben, hatte ich die Idee, selber etwas Ähnliches zu programmieren. Um nicht eine simple Kopie der anderen Website zu machen, habe ich den Fokus auf die Physik und nicht auf Spiele gelegt.

Zusammenfassung

Meine Arbeit umfasst das Programmieren einer Website, auf der man sich möglichst einfach eine zweidimensionale Physiksimulation zusammenbauen, speichern und abspielen kann. Als Physikengine nutze ich JBox2D[2]. Dazu programmierte ich eine Website und zwei Java-Applets, eines zum Erstellen einer Simulation und eines zum Abspielen. Die Website ist in PHP geschrieben, eine Sprache, welche auf fast allen Servern läuft.

1 Die erwähnte Website ist PlayCrafter (http://www.playcrafter.com)

2 JBox2D (http://www.jbox2d.org) ist der Java-Port von Box2D (http://www.box2d.org)

Theorie

Java

Der Hauptteil meines Programms ist in Java geschrieben, deshalb werde ich hier eine kurze Einführung in Java geben sowie einige für meine Arbeit wichtigen Aspekte erläutern.

Java ist eine objektorientierte Programmiersprache. Das heisst, dass Java-Code in Klassen organisiert ist. Eine Klasse ist eine Sammlung von sogenannten Attributen (Variablen) und Methoden (Funktionen), welche meist etwas mit den Attributen zu tun haben. Eine Klasse Fahrzeug kann z.B. die Attribute Farbe, Marke und Gewicht haben, und die Methoden `starten()`, `schalten(Gang)` und `bremsen()`. Klassen können meist instantiiert werden, das heisst, dass ein Objekt der Klasse erstellt wird. Beim Beispiel könnte man mit `new Fahrzeug()` ein neues Fahrzeug bauen. Abstrakte Klassen wie Mathematik können nicht instantiiert werden, da sie meist nur als Funktionssammlung oder als Grundgerüst für andere Klassen dienen. Klassen sind immer Unterklassen einer anderen Klasse und erben deren Methoden und Attribute. Eine neue Klasse Auto, welche als Unterklasse von Fahrzeug definiert ist, kann somit bereits fahren, ohne dass man etwas hätte programmieren müssen. Klassen, für welche keine sog. Superklasse definiert ist, erben in Java automatisch von Object. Object ist die einzige Klasse, welche von keiner anderen Klasse abstammt.

Java ist architekturneutral, das heisst, dass Programme, welche in Java geschrieben sind, auf allen Betriebssystemen und Rechnerarchitekturen gleich laufen. Um das zu erreichen, wird Java-Quellcode in sog. Bytecode kompiliert[3]. Bytecode ist eine Art Zwischenstufe zwischen Quellcode und Maschinencode und wird von einer virtuellen Maschine (bei Java die JVM[4]) während der Programmausführung gelesen und in für das jeweilige Betriebssystem und Computer verständlichen Maschinencode umgewandelt und ausgeführt. Die JVM gibt es für viele bekannte Betriebssysteme wie Windows, Mac OS X, Linux und Solaris.

Java-Programme können entweder als Anwendungen oder als Applets erstellt werden. Eine Anwendung ist ein normales Programm, welches man z.B. vom Desktop aus startet. Ein Applet ist ein eingeschränktes Programm, welches im Browser läuft, ähnlich wie Flash. Damit ein Applet auf das Dateisystem des Computers zugreifen kann, muss es zertifiziert werden, und der Benutzer muss dem Zertifikat zustimmen. Ein Applet kann auch keine neuen Fenster öffnen, sondern hat nur einen bestimmten Bereich im Browserfenster zur Verfügung. Applets sind immer Unterklassen der Klasse Applet.

3 Beim Kompilieren wird für Menschen verständlicher Quellcode in für Computer verständlichen Maschinencode, oder wie hier in Bytecode, umgewandelt.

4 **J**ava **V**irtual **M**achine

Variablen

Eine Variable, welche nicht von einem nativen Typ[5] ist, ist in Java immer eine Referenz (auch Zeiger oder Pointer genannt) auf den eigentlichen Speicherort der Variable. Das verhindert, dass man versehentlich Funktionen den Wert einer grossen Variable, z.B. eines Bildes, übergibt, sondern nur die Referenz zum Bild.

Wird eine nicht-native Variable mit dem Zuweisungsoperator (`=`) einer anderen Variable zugewiesen, wird nicht der Wert der Variable kopiert, sondern nur die Referenz, sodass beide Variablen auf den gleichen Speicherort verweisen. Will man den Wert der Variable kopieren, muss die Klasse der Variable klonbar sein (d.h. sie muss das Interface 'Cloneable' implementieren), und es muss die `clone()`-Funktion benutzt werden, welche den Wert der Variable kopiert und eine Referenz zur Kopie zurückgibt. Arrays können ebenfalls mit `clone()` kopiert werden.

Schnittstellen

Schnittstellen (Interfaces) sind spezielle Klassen, welche Methoden und Attribute definieren, wobei die Methoden wie bei abstrakten Methoden nicht implementiert werden und die Attribute keine Werte haben. Interfaces können von anderen Klassen implementiert werden, was bedeutet, dass sie alle Methoden implementieren und die Attribute definieren. Eine Klasse kann beliebig viele Interfaces implementieren.

Threads

Java unterstützt Multithreading, was bedeutet, dass ein Java-Programm in mehreren Threads ablaufen kann. Threads sind voneinander mehr oder weniger unabhängige Teile eines Programms, welche meist gleichzeitig ausgeführt werden. Bei Computern mit mehreren Prozessoren ist Multithreading besonders effizient, da Threads auf verschiedenen Prozessoren ausgeführt werden können, was die Leistung deutlich verbessern kann.

Threads haben allerdings auch den Nachteil, dass man nie weiss, wann genau ein Thread ausgeführt wird. Dies erschwert die Kommunikation zwischen Threads deutlich.

Schlüsselwörter

Java hat, wie jede Programmiersprache, Schlüsselwörter, welche eine spezielle Bedeutung haben. Einige sind von anderen Programmiersprachen bekannt, andere wiederum sind komplett neu und können beim Programmieren sehr nützlich sein. Hier ist eine Liste von wichtigen Java-Schlüsselwörtern:

`private`, <ohne>[6], **`protected`**, **`public`**: Zugriffsmodifikatoren, sie schränken also den Zugriff auf das folgende Element ein, welches ein Attribut oder eine Methode sein kann. Klassen können nur **`public`** oder nichts haben. Private Elemente stehen nur der Klasse selbst zur Verfügung, ohne Modifizierer sind sie für das ganze Paket sichtbar, mit **`protected`** können auch Unterklassen darauf zugreifen und **`public`** macht Elemente für alle Klassen sichtbar.

5 Die nativen Typen von Java sind: `byte`, `short`, `int`, `long`, `float`, `double`, `boolean` und `char`.

6 Wird auch default oder package private genannt, welches jedoch keine Schlüsselwörter sind.

`static`: Auf das bezeichnete Element kann ohne Objektreferenz zugegriffen werden, man greift mithilfe der Klasse darauf zu. Das bedeutet, dass der Wert nicht in jeder Instanz der Klasse vorhanden ist, sondern im ganzen Programm genau einmal. Ein solches Element entspricht also etwa globalen Variablen in anderen Programmiersprachen.

`final`: Eine finale Variable kann ihren Wert nicht ändern, eine finale Methode kann nicht überschrieben werden, und eine finale Klasse kann nicht als Superklasse benutzt werden. Dieses Schlüsselwort ist zur Codeoptimierung gedacht, da der Compiler Variablenzugriffe direkt machen kann, und nicht über eine Referenz auf den Wert der Variable zugreifen muss. Es ist deshalb ein sehr nützliches Schlüsselwort für Funktionsargumente, da man deren Wert fast nie verändert. Es hilft somit auch Fehler zu vermeiden, da der Compiler einen Fehler ausgibt, wenn man das Argument in der Funktion verändert.

`abstract`: Klassen mit diesem Bezeichner können nicht instantiiert werden, und abstrakte Methoden, welche nur in abstrakten Klassen vorkommen können, müssen von Unterklassen erst noch implementiert werden.

`synchronized`: Bezeichnet Elemente, auf welche nur ein Thread gleichzeitig zugreifen kann. Wird auf das Element zugegriffen, wird es automatisch für sämtliche Threads gesperrt, bis der darauf zugreifende Thread es wieder freigibt.

`assert`: Ist kein Modifikator wie die anderen aufgelisteten Schlüsselwörter, sondern eine Anweisung, welche einen Fehler erzeugt, falls der nachfolgende Ausdruck falsch ist. Assert kann, wenn man genug davon an sinnvollen Stellen im Code platziert, zur Fehlerbehebung sehr nützlich sei.

`instanceof`: Ist ebenfalls kein Modifikator, sondern ein Operator. Er prüft, ob ein Objekt Instanz der angegebenen Klasse oder einer derer Unterklassen ist.

Aufzählungen

Aufzählungen, auf Englisch enumerations, sind Pseudo-Klassen, in welchen eine Aufzählung von verschiedenen Konstanten gemacht wird. Diesen Konstanten können Werte zugeordnet werden, auf welche dann durch die Konstanten zugegriffen werden kann. Als Beispiel zeige ich hier einen kurzen Ausschnitt aus der Klasse Strings, welche sämtliche übersetzbaren Zeichenketten enthält. Jeder Konstanten wird hier je ein englischer und ein deutscher String zugeordnet, welche gleich in der richtigen Sprache von der Funktion `toString()` zurückgegeben werden. Das `@Override` bedeutet, dass die Implementierung dieser Funktion aus der Superklasse überschrieben wird, welche in diesem Beispiel LIBRARY, CANVAS bzw. KIT ausgeben würde, statt der erwünschten Übersetzung. Funktionen in Aufzählungen werden mithilfe der Konstanten aufgerufen, zum Beispiel erhält man die Übersetzung vom Baukasten mit `KIT.toString()`, oder in diesem speziellen Fall auch mit `""+KIT`.

```
public enum Strings {
    // Dies sind die Konstanten und die zugehörigen Werte
    LIBRARY  ("Library", "Bibliothek"),
    CANVAS ("Canvas", "Leinwand"),
    KIT ("Kit", "Bausatz");

    // in dieser Variablen werden die Werte gespeichert
    private final String[] s = new String[2];

    // Konstruktor. Definiert, wie die Werte gespeichert werden sollen
    private Strings(String en, String de) {
        s[0] = en;
        s[1] = de;
    }

    // gibt den String in der passenden Sprache aus
    // Config.lang ist entweder 0 (Englisch) oder 1 (Deutsch)
    @Override
    public final String toString() {
        return s[Config.lang];
    }
}
```

Rendern

Hardwarebeschleunigung

Die meisten Zeichen- und Bildoperationen in Java werden auf dem Prozessor ausgeführt, und nicht auf der Grafikkarte. Der Prozessor ist jedoch nicht dafür geeignet, und deshalb sind diese Operationen sehr langsam. Die Klasse VolatileImage ist eine Möglichkeit, die Grafikkarte zum Rendern zu benutzen. Bilder dieser Klasse haben allerdings auch den Nachteil, dass sie aus dem Speicher verschwinden, wenn eine andere Anwendung im Vollbildmodus ausgeführt wird, und dazu reicht bereits der Bildschirmschoner. Ich benutze solche Bilder zur Darstellung der Simulation.

Besseres Zeichnen

Die standardmässige Klasse, um auf den Bildschirm oder auf Bilder zu zeichnen, ist die Klasse Graphics. Diese beinhaltet jedoch nur sehr wenige Funktionen, welche dazu noch verpixelt zeichnen. Als Alternative bietet sich die Klasse Graphics2D an, welche als Unterklasse von Graphics überall dort eingesetzt werden kann, wo auch Graphics eingesetzt werden kann. Sie überschreibt alle Funktionen von Graphics und erweitert sie um Gleitkommazahl-Argumente, teilweise auch um Hardwarebeschleunigung, sowie um sogenannte 'Rendering Hints'. Mit diesen kann unter anderem Interpolation und Antialiasing benutzt werden.

PHP

PHP (**P**HP: **H**ypertext **P**reprocessor, ursprünglich **P**ersonal **H**ome **P**age Tools) ist eine serverseitige Skriptsprache, läuft also auf einem Server. Beim Aufruf eines PHP-Skripts durch einen Browser wird das Skript ausgeführt und dessen Ausgabe an den Browser gesendet. Deshalb wird PHP oft zur Erstellung von dynamischen Webseiten in HTML genutzt, es können jedoch auch andere Inhalte generiert werden, wie z.B. PDF-Dateien oder Bilder.

Da PHP auf sehr vielen Servern läuft und viele verschiedene Datenbanken unterstützt, habe ich meine Webseite in PHP geschrieben.

MySQL

MySQL ist ein Datenbanktyp. Die Anfragen an die Datenbank basieren auf SQL (**S**tructured **Q**uery **L**anguage, engl. für strukturierte Abfragesprache), was bedeutet, dass die ganze Interaktion in Textform abläuft. MySQL ist deshalb sehr einfach zu benutzen, allerdings sind die Anfragen auch Sicherheitslücken, wenn man sich nicht gegen Angriffe schützt. MySQL ist wie PHP weit verbreitet, und auf vielen Servern die einzige verfügbare Datenbank.

XML

XML (E**x**tensible **M**arkup **L**anguage, engl. für erweiterbare Auszeichnungssprache) ist, wie der Name bereits sagt, eine Auszeichnungssprache. Solche Sprachen sind keine Programmiersprachen, sondern dienen der Beschreibung von Daten. In meinem Programm werden mit XML Simulationen und Bauteile gespeichert, was z.B. so aussieht:

```
<piece>
    <shapes>
        <box width="0.4" height="1.8"/>
    </shapes>
    <physics>
        <rotation fixed="true"/>
    </physics>
    <visual layer="8" color="#de7834"/>
</piece>
```

Praktischer Teil

Idee

Meine Anwendung besteht aus zwei Teilen, einem Editor-Applet und einem Abspiel-Applet, welche allerdings teilweise auf demselben Code basieren.

Der Editor beinhaltet eine Bibliothek, in der alle verfügbaren Bauteile aufgelistet sind. Die Beschreibungen der Bauteile werden aus einer XML-Datei gelesen, und die Vorschaubilder werden heruntergeladen. Die Bauteile können zum Bausatz hinzugefügt werden, welcher alle in der Simulation benutzten Bauteile enthält. Die Trennung von der Bibliothek und dem Bausatz ist vor allem bei einer grossen Anzahl von Bauteilen nützlich, damit man nur die, welche man braucht, im Bausatz hat, und nicht immer die ganze Bibliothek durchsuchen muss, und ausserdem werden nur die Bauteile des Bausatzes vollständig heruntergeladen. Dieses Konzept von einer Bibliothek und einem Bausatz stammt wie die Idee ebenfalls von PlayCrafter. Die Bauteile im Bausatz können mittels Drag&Drop auf die Leinwand gezogen werden, welche die physikalische Welt der Simulation enthält. Es können diverse Einstellungen an der Welt vorgenommen werden, und die physikalischen Eigenschaften der Bauteile im Bausatz können angepasst werden, sodass gleiche Bauteile auch gleiche Eigenschaften haben. Die Physik der Bauteile ist vererbbar: Die Simulation hat Standardwerte, welche von Bauteilen im Bausatz überschrieben werden können, und diese wiederum von den eigentlichen Bauteilen auf der Leinwand. So können leicht Anpassungen an vielen Bauteilen gleichzeitig durchgeführt werden.

Die Simulation kann jederzeit getestet werden. Beim Testen wird eine neue Instanz der Simulation-Klasse generiert. Dieser werden sämtliche Informationen über die Welt und die Bauteile übergeben, welche sie dann in ein für die Physikengine JBox2D verständliches Format konvertiert. Dann lässt sie die Physikengine in regelmässigen Abständen Schritt für Schritt die Simulation durchrechnen und stellt diese jeweils grafisch dar, wozu hardwarebeschleunigte VolatileImages verwendet werden.

Wird die Simulation gespeichert, wird eine XML-Datei erstellt, in welcher alle Bauteile und die Welt beschrieben sind. Diese wird mitsamt Screenshot mittels HttpComponents[7] auf der Webserver geladen, wo sie ein PHP-Skript in eine Datenbank schreibt.

Das Abspiel-Applet ist viel einfacher aufgebaut: Es liest die XML-Datei der als Parameter übergebenen Simulation, lädt alle erforderlichen Bauteile vom Server herunter und strukturiert sie entsprechend der Simulation. Dann wird wie im Editor eine neue Simulation-Instanz generiert und diese erledigt den Rest.

Website

```
http://www.physicssite.ch.vu/
```

Für mein Applet habe ich eine Website entworfen, welche in PHP programmiert ist. Auf der Website können Simulationen erstellt werden, es können aber auch Simulationen von anderen abgespielt werden, welche entweder in der Liste aller Simulationen oder mithilfe einer Suchfunktion gefunden werden. Auf dem Server befinden sich alle PHP-Skripte, das

7 Eine API von Apache, welche die Benutzung von HTTP stark vereinfacht (http://hc.apache.org/)

Programm in stabiler und Beta-Version, sowie eine MySQL-Datenbank, in welcher alle Bauteile, Simulationen und deren Bilder gespeichert sind. Ausserdem hat es ein PHPBB[8]-Forum, auf dem man Feedback geben kann und ich alle Änderungen, welche ich an der stabilen Programmversion mache, aufliste. Es dient auch dazu, dass man sich als Benutzer anmelden kann und die erstellten Simulationen dann unter seinem Benutzernamen gespeichert werden.

Die Website wird von 000webhost.com kostenlos gehostet, und die Domain, welche ebenfalls kostenlos ist, stammt von nic.ch.vu.

Sprache

Ich habe das ganze Programm nicht nur in Deutsch entworfen, sondern auch in Englisch, da ich es im Internet veröffentliche. Die Website ist ebenfalls zweisprachig, und die Sprache kann leicht umgeschaltet werden. Die Sprache wird als Parameter an das Applet übergeben, welches dann alle Strings in der jeweiligen Sprachen benutzt. Dazu wird die Klasse Strings, eine Aufzählung (Enumeration), verwendet, welche für jeden Wert zwei Strings enthält, von welchen je der Passende ausgegeben wird.

Auf der Website sind sämtliche Strings in zwei grossen Arrays enthalten. Wird ein PHP-Skript ausgeführt, verwendet es das Array in der Sprache des Benutzers.

Codedaten

Ich habe das ganze Programm selber programmiert, bis auf die Berechnung der Physik, was die Physikengine JBox2D übernimmt, sowie den HTTP-Upload, welchen Apaches HttpComponents erledigt.

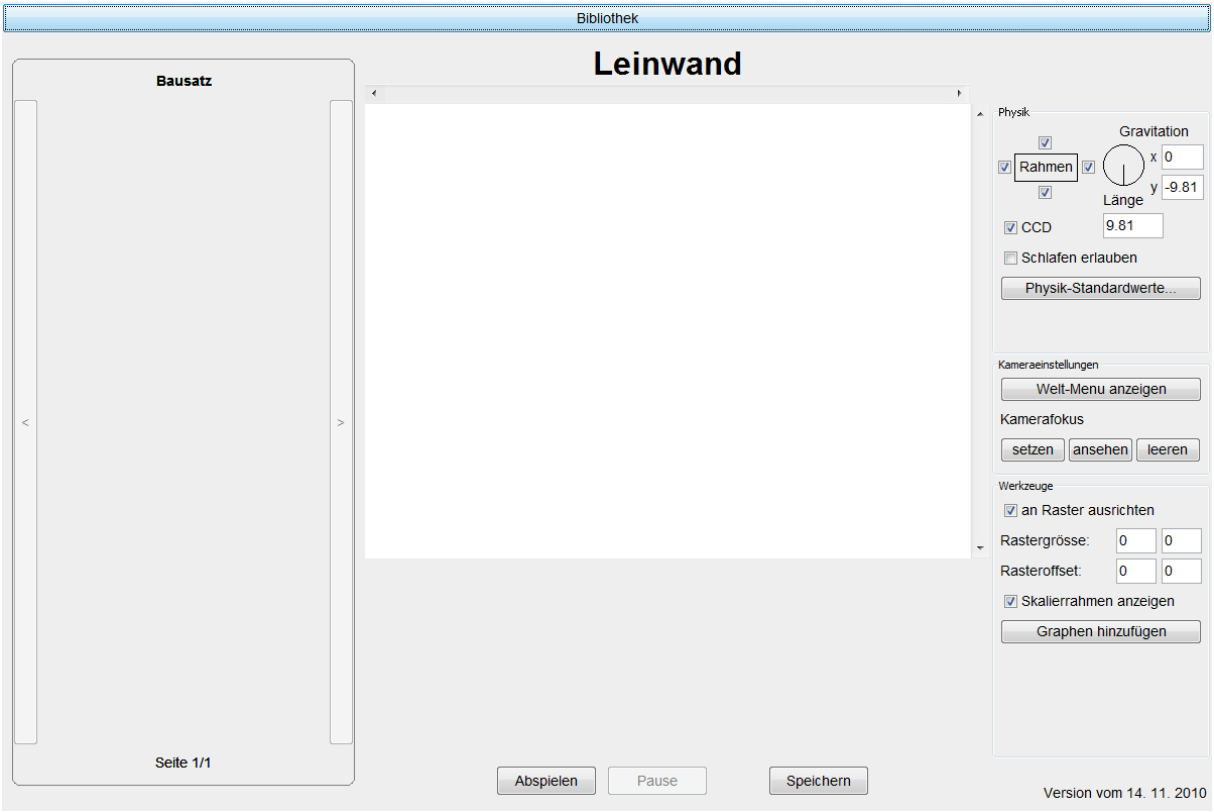
Insgesamt habe ich rund 6'500 Zeilen Pragrammcode mit 200'000 Zeichen geschrieben, wovon jedoch knapp 10% (625 Zeilen) Imports sind, welche die Entwicklungsumgebung automatisch hinzugefügt hat.

Der Code ist in dieser Arbeit nicht enthalten, da er sich noch ändern wird. Er kann aber von der Webseite aus heruntergeladen werden.

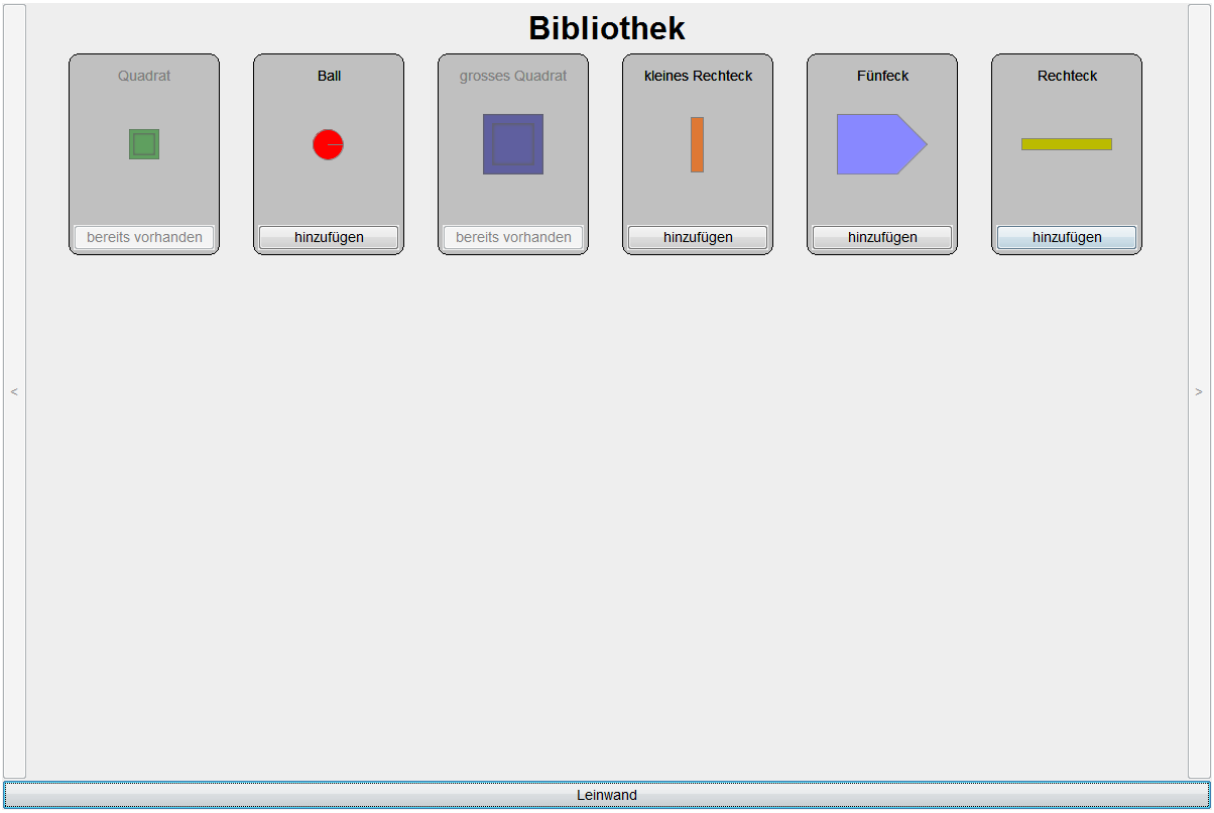
8 in PHP programmiertes Forum, http://www.phpbb.com/

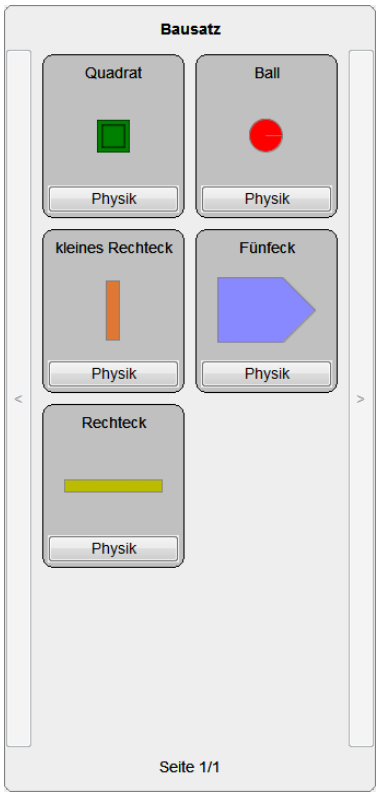
Manual

Startet man das Applet, sieht man folgendes auf dem Bildschirm:



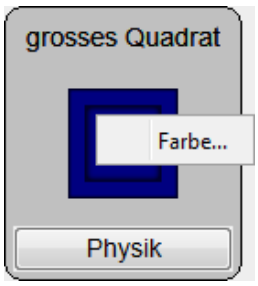
- Die breite Schaltfläche ganz oben im Bildschirm führt zur Bibliothek.
- Der mit Bausatz bezeichnete Bereich links beinhaltet alle Bauteile, welche der Benutzer aus der Bibliothek auswählt.
- Die Leinwand in der Mitte stellt die physikalische Welt mit allen hinzugefügten Bauteilen dar.
- Rechts kann man verschiedene Werte der Welt einstellen, namentlich die Physik sowie die Kamera, ausserdem hat es einige nützliche Werkzeuge für das Zusammenbauen der Simulation.
- Unten hat es Schaltflächen zum Testen und Speichern der Simulation.
- Rechts unten ist die Versionsanzeige.



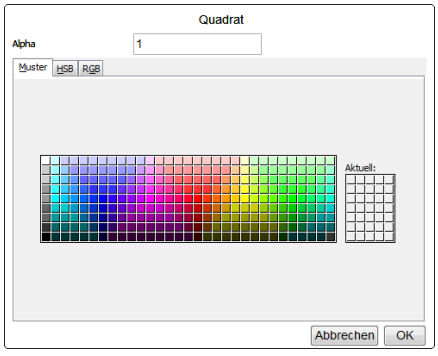


Die **Bibliothek** enthält alle Bauteile, welche sich aus dem Server befinden. Man kann ein Bauteil durch Klicken auf 'hinzufügen' zum Bausatz hinzufügen, was einige Zeit dauern kann, da das Bauteil erst noch heruntergeladen werden muss.

Der **Bausatz** enthält die hinzugefügten Bauteile aus der Bibliothek. Hier können die Farbe und die physikalischen Eigenschaften der Bauteile angepasst werden. Per Drag&Drop auf die Leinwand werden sie der Simulation hinzugefügt.

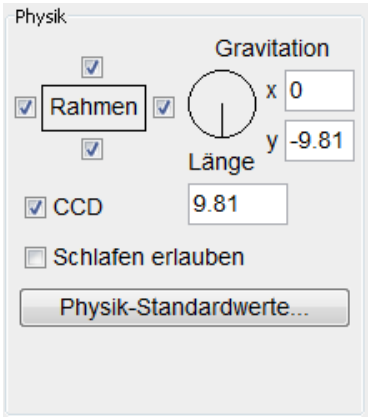


Durch Rechtsklick auf ein Bausatzteil kommt man zum **Farbmenü**, wo die Farbe und Transparenz des Bauteils festgelegt werden kann.



Die **Leinwand** stellt die zusammengebaute Simulation dar. Hier können Bauteile hinzugefügt, ausgewählt, gelöscht, kopiert, gedreht, verschoben und skaliert werden. Einzelne Bauteile werden ganz einfach mit einem Mausklick ausgewählt, mehrere können durch Klicken & Ziehen der Maus ausgewählt werden. Die ausgewählten Bauteile können mit der linken Maustaste verschoben und mit der Rechten gedreht werden. Gedreht wird um den Mittelpunkt der Bauteile, nicht um ihren Schwerpunkt. Klickt man auf einen der Eckpunkte des Auswahlrechtecks, kann man die ausgewählten Bauteile skalieren.

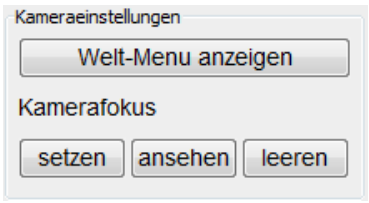
Ausgewählte Bauteile können mit Ctrl+C kopiert, mit Ctrl+X ausgeschnitten und mit Ctrl+V eingefügt werden. Gelöscht werden Bauteile mit delete. Alle Bauteile können mit Ctrl+A ausgewählt werden, und ausgewählte Bauteile können mit der Leertaste angezeigt werden.



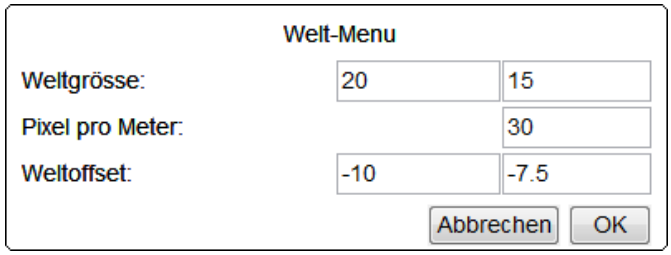
Im Feld **Physik** können Einstellungen an der Physik der Simulation vorgenommen werden. Man kann die Gravitation einstellen, ob und wo es einen Rahmen um den sichtbaren Bereich gibt, die Physik-Standardwerte sowie spezielle Einstellungen der Physikengine:

CCD (continous collision detection): Eine Technik, welche verhindert, dass Bauteile in andere Bauteile eindringen können. Sie erhöht somit die Genauigkeit der Simulation, ist jedoch sehr rechenintensiv und kann die Simulation stark verlangsamen.

Schlafen erlauben: verhindert, dass sich sehr langsame Objekte bewegen, indem die Geschwindigkeit, sobald sie einen bestimmten Wert unterschreitet, auf Null gesetzt wird. Kann für Idealsimulationen verwendet werden, in denen z.B. zwei gleiche Massen vollkommen unelastisch zusammenprallen, wodurch sie eigentlich stehen bleiben sollten. Jedoch bewegen sie sich aufgrund von Rundungsfehlern trotzdem ein wenig, was mit dieser Option unterbunden werden kann.

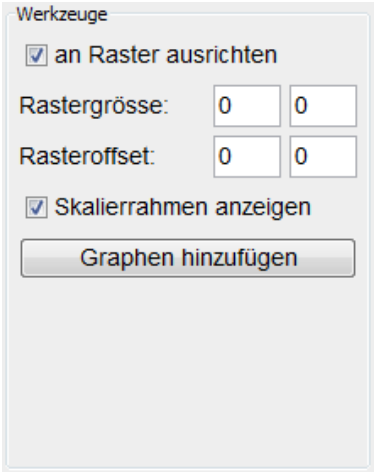


Unter **Kameraeinstellungen** kann das Welt-Menü angezeigt werden, in welchem man die Weltgrösse, den Zoom in Pixeln pro Meter sowie den Offset des Nullpunktes einstellen kann. Unmögliche Eingaben werden automatisch korrigiert, sodass man keine Karte machen kann, welche kleiner als der Bildschirm wäre.



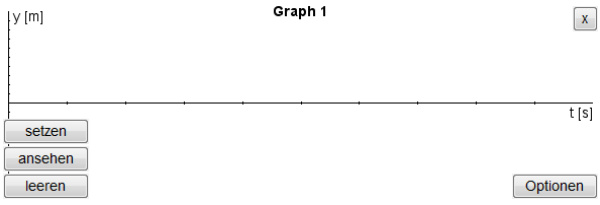
Ausserdem kann man ein Bauteil oder mehrere Bauteile bestimmen, welchen die Kamera folgen soll, falls nicht die ganze Welt auf dem Bildschirm Platz hat. Diese Bauteile werden durch Klicken auf 'setzen' auf die aktuell ausgewählten Bauteile auf der Leinwand gesetzt. Sind mehrere Bauteile ausgewählt, folgt die Kamera ihrem Schwerpunkt.

Bei den **Werkzeugen** findet man Einstellungen zum Platzieren und Verschieben von Bauteilen an einem Raster. Man kann die Rastergrösse festlegen sowie den Offset zur linken unteren Ecke der Welt. Ausserdem kann man den Auswahlrahmen ausschalten, falls er stört.

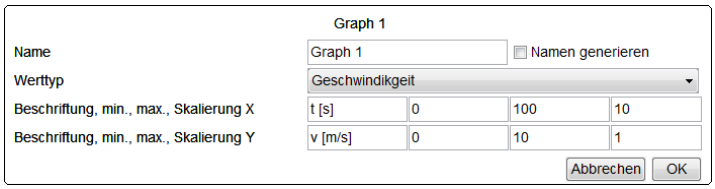


Man kann auch einen oder mehrere **Graphen** hinzufügen, welche je einen bestimmten Wert wie die Geschwindigkeit oder die Energie eines oder mehrerer Bauteile darstellen. Diese Bauteile werden wie bei der Kamera bestimmt, indem man die gewünschten Bauteile auf der Leinwand auswählt und dann im Graphen auf setzen klickt.

Durch Klick auf 'Optionen' erscheint das Optionsmenü des Graphen, wo man dessen Namen ändern oder ihn automatisch aufgrund des Werttyps und der ausgewählten Bauteile generieren lassen kann. Weiter kann man den Werttyp einstellen, sowie für jede Achse die Beschriftung, den minimalen und maximalen Wert und die Unterteilung.

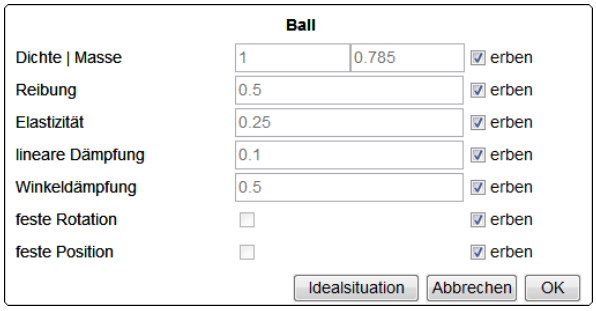


Der Werttyp kann z.B. die Position, die Geschwindigkeit oder die Energie der Bauteile sein. Bei mehreren Bauteile gibt es Spezialfälle: Die Position entspricht dem Schwerpunkt, die Geschwindigkeit ist die Geschwindigkeit des Schwerpunktes und die Drehgeschwindigkeit ist der nach den Massen gewichtete Durchschnitt aller Drehgeschwindigkeiten.



Einstellungen, welche in mehr als 20 Unterteilungen einer Achse resultieren würden, werden verhindert, um die Achsen übersichtlich zu halten.

Im **Physik-Menü** kann man physikalische Einstellungen machen. Man kann es durch die Schaltfläche 'Physik-Standardwerte' im Feld Physik (rechts oben im Bildschirm), durch die Schaltfläche 'Physik' bei irgendeinem Bausatzteil, oder per Rechtsklick auf ein Leinwand-Bauteil öffnen.

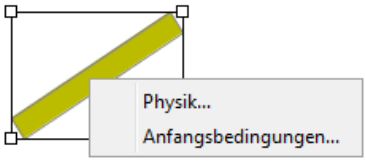


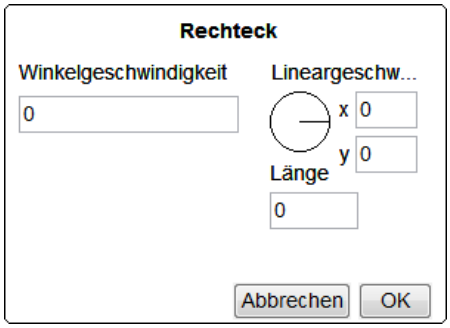
Die möglichen Einstellungen sind:

- Dichte / Masse: Die Dichte bzw. Masse des Bauteils. Wird eines von beiden eingegeben, wird das andere automatisch ausgerechnet, gespeichert wird jedoch nur die Dichte. Wird also ein Bauteil vergrössert oder verkleinert, muss man die Masse neu einstellen. Vererbt wird ebenfalls nur die Dichte.
- Reibung: Der Reibungskoeffizient.
- Elastizität: Je höher dieser Wert, desto mehr Energie bleibt bei einer Kollision erhalten, wobei die Werte beider kollidierender Teile beachtet werden.
- lineare Dämpfung: Betrag, um den die Geschwindigkeit pro Sekunde verringert wird.
- Winkeldämpfung: Betrag, um den die Winkelgeschwindigkeit pro Sekunde verringert wird.
- feste Rotation: Verhindert, dass sich das Bauteil dreht.
- feste Position: Verhindert jegliche Bewegung des Bauteils.

Alle physikalischen Einstellungen können vom jeweils übergeordneten Objekt geerbt werden, wobei die Simulation das oberste Objekt ist und somit die Standardwerte für alle Bauteile definiert.

Im Kontextmenü eines Bauteiles auf der Leinwand gibt es ausserdem die Option **Anfangsbedingungen**, welche ein Menü öffnet, in dem die anfängliche lineare Geschwindigkeit und die Winkelgeschwindigkeit eingestellt werden können.





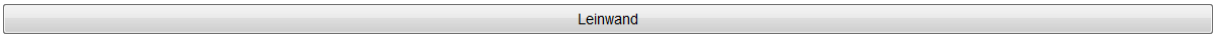
Beispiel eines Simulationsaufbaus

Zur besseren Verständlichkeit des Programms wird hier gezeigt, wie man eine einfache Simulation zusammenbauen kann. Als Beispiel wird ein Ball, welcher eine Ebene hinunter rollt, verwendet.

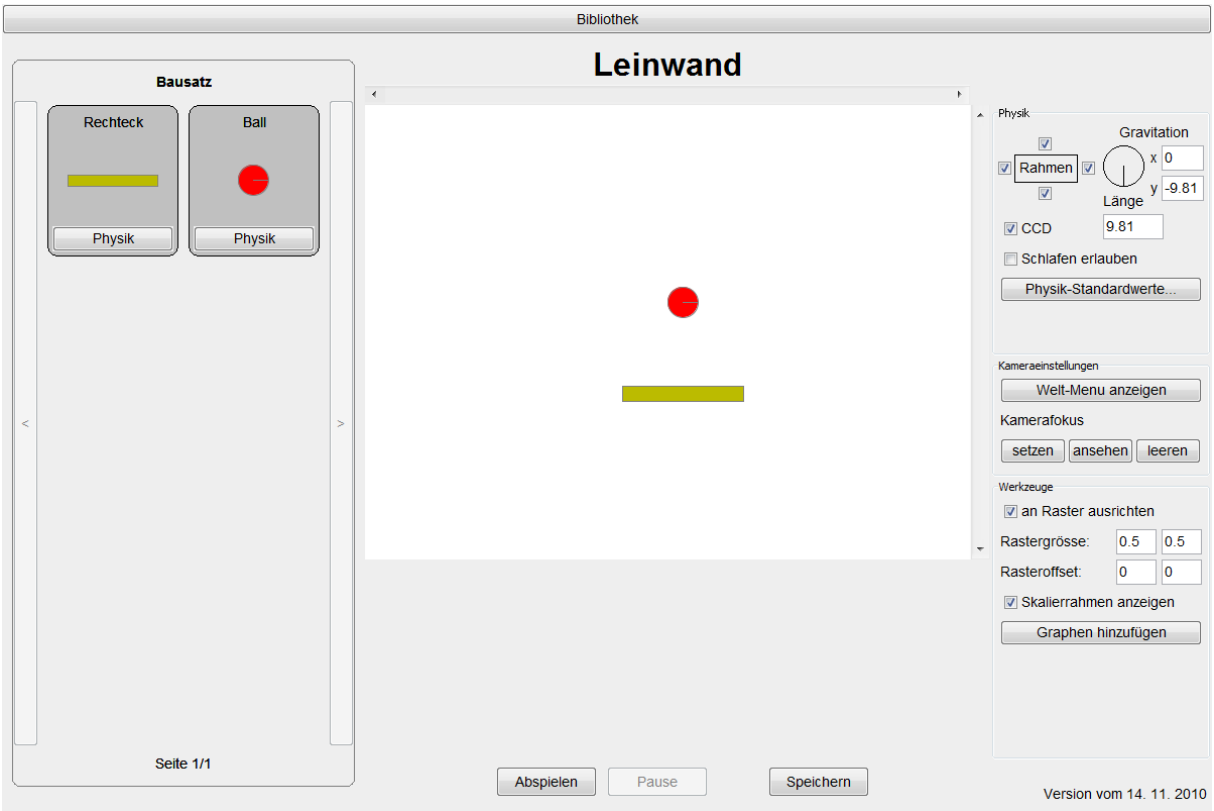
Als allererstes muss man Bauteile zum Bausatz hinzufügen. Dazu geht man über die grosse Schaltfläche am oberen Rand des Bildschirms zur Bibliothek:

Dort kann man nun die Bauteile auswählen, die man braucht, und sie durch Klicken auf die jeweilige Schaltfläche 

zum Bausatz hinzufügen. Für dieses Beispiel nimmt man den Ball und das gelbe Rechteck. Um wieder aus der Bibliothek raus zu kommen, klickt man auf der grosse Schaltfläche ganz unten im Bildschirm:

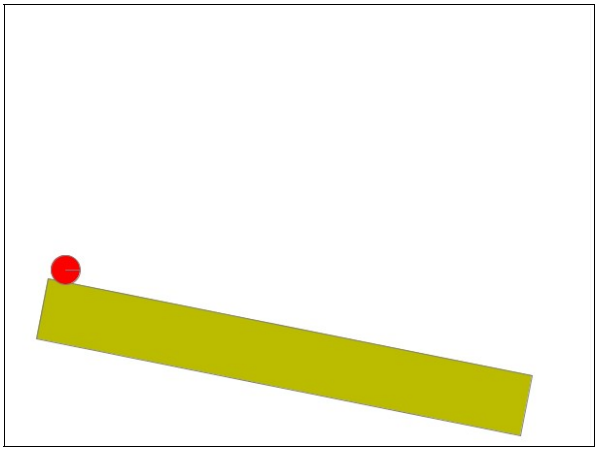


Als nächstes kann man die Bauteile mit der Maus auf die Leinwand ziehen. Das Ganze sollte nun so aussehen:

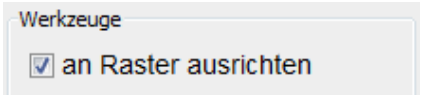


Nun wird das Rechteck zu einer Ebene gemacht. Dazu klickt man es an, wodurch es wie folgt aussieht: 

Um das Rechteck zu vergrössern, klickt man auf eines der vier weissen Quadrate und zieht es nach aussen. Dann dreht man das vergrösserte Rechteck, indem man es mit der rechten Maustaste irgendwo anklickt und die Maus in die gewünschte Richtung zieht, am besten so, dass das Rechteck eine leichte Neigung nach rechts hat. Danach verschiebt man den Ball an den Anfang der schiefen Ebene, sodass man folgendes Bild erhält:

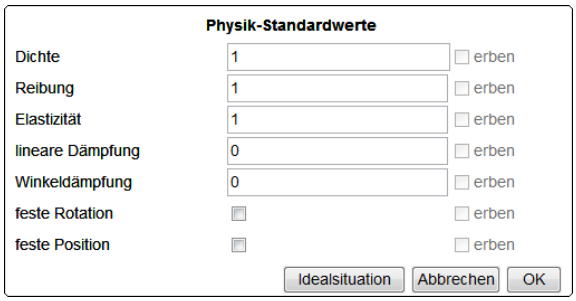


Jetzt befindet sich der Ball aber noch nicht genau auf der Ebene. Will man ihn besser positionieren können, muss man die Option 'an Raster ausrichten' ausschalten, welche sich im rechten Bereich des Bildschirmes befindet:



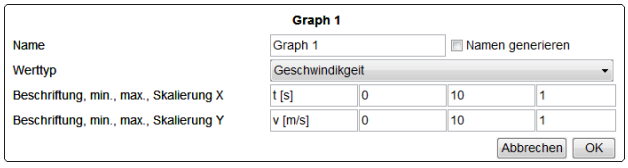
Um die Simulation noch weiter zu vereinfachen, sollte man die Physik der Simulation möglichst einfach halten. Dafür klickt man auf die Schaltfläche [Physik-Standardwerte...], welche sich ebenfalls rechts im Bildschirm befindet.

Im daraufhin erscheinenden Fenster klickt man auf 'Idealsituation', um eine idealisierte Situation zu bekommen. Allerdings würde sich in solch einer Situation mit keiner Reibung unser Ball nicht drehen, sondern einfach die Ebene herunter gleiten. Deshalb geben wir in die Spalte Reibung den Wert 1 ein. Jetzt sollte das Menü folgendermassen aussehen:



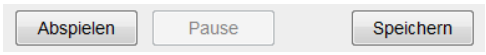
Das Menü wird mit Klick auf OK geschlossen.

Jetzt kann zur Darstellung der Geschwindigkeit des Balls noch ein Graph hinzugefügt werden. Dazu klickt man auf die Schaltfläche [Graphen hinzufügen]. Im neuen Graphen klickt man auf [Optionen], und stellt die Zeit auf 10 Sekunden und die Skalierung auf 1 ein, da der Ball sicher nicht länger als 10 Sekunden braucht, um die Ebene herunterzurollen. Dieses Menü schliesst man ebenfalls mit OK.

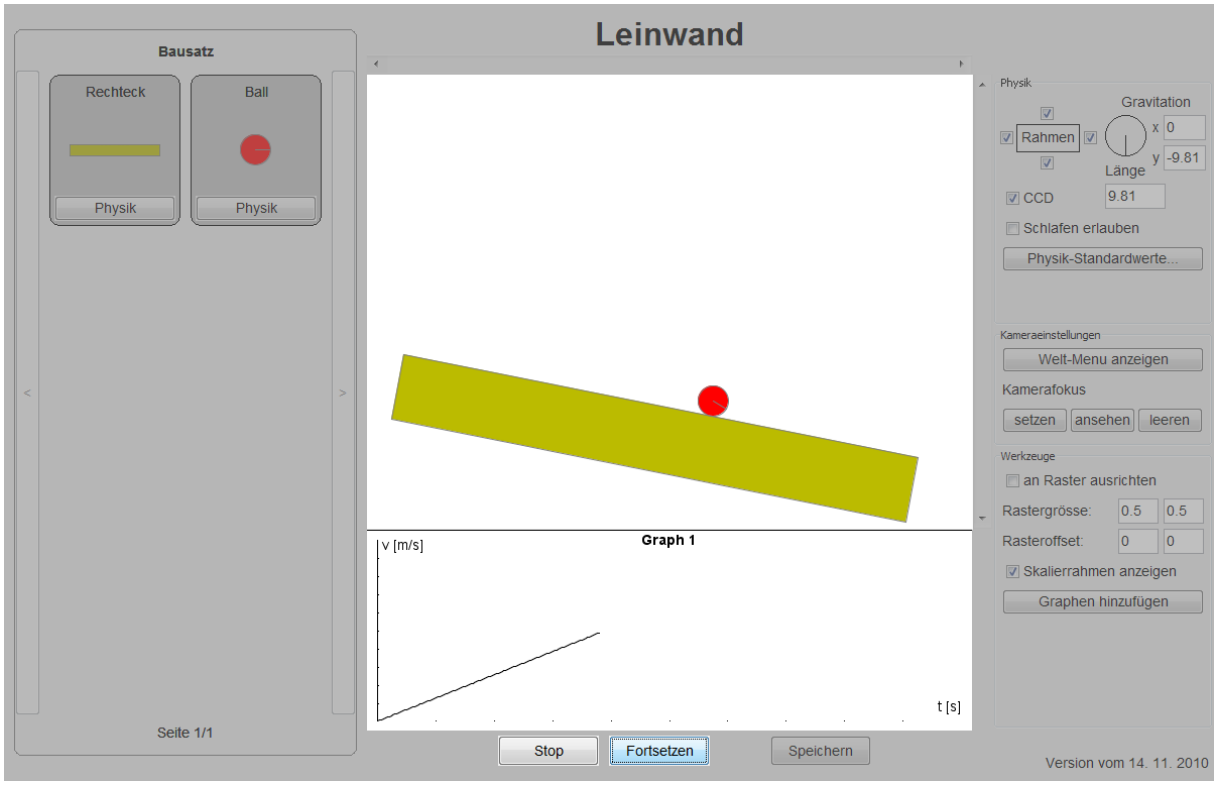


Schlussendlich muss man dem Graph noch sagen, von welchem Bauteil er die Geschwindigkeit nehmen soll. Dazu wählt man den Ball auf der Leinwand aus und klickt dann auf die Schaltfläche [setzen].

Nun kann man die Simulation durch einen Klick auf 'Abspielen' laufen lassen, und mit 'Speichern' kann man die Simulation speichern.

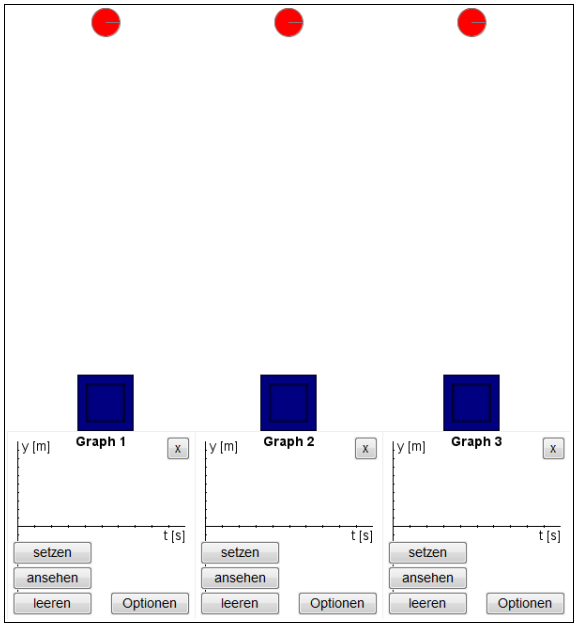


Wenn man die Simulation laufen lässt, erhält man folgendes Bild:

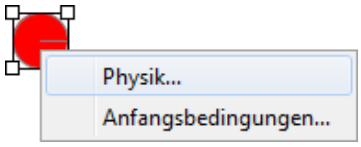


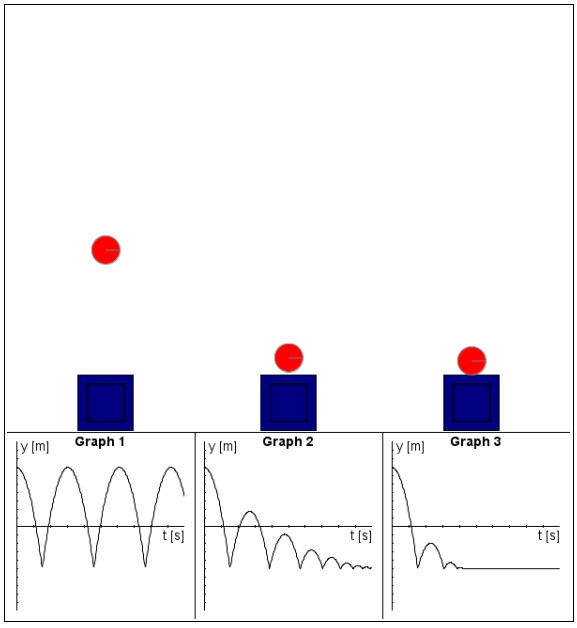
Noch ein Beispiel

Man startet wieder wie vorhin, nur wird hier das grosse Quadrat anstelle des Rechtecks ausgewählt. Auch die Physik wird wieder idealisiert, wobei die Reibung dieses Mal keine Rolle spielt. Dann platziert man ein Quadrat sowie einen Ball weit darüber, wählt die beiden Bauteile aus und drückt Ctrl+C und danach Ctrl+V. Damit hat man die Bauteile kopiert. Achtung: die kopierten Bauteile liegen genau über den bisherigen Bauteilen, sie sollten also gleich wegbewegt werden. Dann drückt man Ctrl+V nochmals und schiebt die neuen Bauteile wiederum weg. Jetzt fügt man noch 3 Graphen hinzu, stellt ihren Typ auf Y-Position, mit Y-Wert von -10 bis 10, und Zeit von 0 bis 10. Schlussendlich wird jedem Graphen noch ein Ball zugeordnet. Nun sollte es wie folgt aussehen:



Damit die 3 Bälle nicht synchron hüpfen, stellt man mittels Rechtsklick auf die Bälle für den zweiten Ball als Elastizität 0.75 ein, und für den Dritten 0.5. Startet man nun die Simulation, sieht man, dass die Bälle trotzdem gleichzeitig springen. Das liegt daran, dass die Quadrate immer noch 100% elastisch sind. Also stellt man für jedes Quadrat den gleichen Wert ein wie für den jeweiligen Ball. Wenn die Simulation jetzt gestartet wird, sieht man, wie erwartet, Folgendes:





Falls erwünscht, kann im Welt-Menü der Y-Offset auf -2 gesetzt werden, sodass der Nullpunkt genau auf dem mittleren Quadrat zu liegen kommt. Das verschiebt jedoch alle Bauteile nach unten, die Quadrate sogar aus dem Bildschirm raus. Um sie wieder in den sichtbaren Bereich zu bekommen, drückt man Ctrl+A, um alle Bauteile auszuwählen. Nun kann man die Auswahl an einem Ball nach oben ziehen, und die Quadrate kommen wieder zum Vorschein. Dann muss man noch die Graphen anpassen, sodass ihr Y-Wert von 0 bis 13 geht, was die ganze Leinwand oberhalb der Quadrate abdeckt.

Eigene Bauteile erstellen

Anmerkung: Dieser Abschnitt befasst sich mit einem Feature des Programms, welches bei der Abgabe der Arbeit noch nicht vorhanden ist, da man eigene Bauteile nur lokal benutzen und nicht auf den Server laden kann.

Sämtliche Bauteile und Simulationen sind im XML-Format gespeichert. Die XML-Datei der Simulation wird vom Editor automatisch generiert und beschreibt den Aufbau der Simulation, namentlich die Bauteile im Bausatz sowie auf der Leinwand, deren Physik und die der Simulation sowie die Kameraeinstellungen und die Graphen.

Die Bauteile jedoch müssen von Hand geschrieben werden, allerdings sind die XML-Dateien auch viel einfacher aufgebaut.

Als Beispiel nehme ich das gelbe, lange Rechteck: 

```
<piece>
    <shapes>
        <box width="4" height="0.5"/>
    </shapes>
    <physics>
        <position fixed="true"/>
    </physics>
    <visual color="#bbbb00"/>
</piece>
```

Das Bauteil (piece) besteht aus nur einer Form (shape): einem Rechteck (box). Es erbt alle physikalischen Werte (physics) bis auf die feste Position (fixed position), welche es auf wahr (true) setzt. Es hat ausserdem eine voreingestellte Farbe (color).

Dies sind nur einige wenige der möglichen Einstellungen. Hier folgt nun eine Liste aller benutzbaren Elemente:

Formen

Es können beliebig viele der folgenden vier Formen für ein Bauteil verwendet werden:

Quadrat / Rechteck

Anmerkung: Die Rotation ist der Winkel in Grad, um welchen das Rechteck im Gegenuhrzeigersinn gedreht wird.

```
<square x="0" y="0" width="1" rotation="0"/>
```

```
<box x="0" y="0" width="1" height="1" rotation="0"/>
```

Kreis

```
<circle x="0" y="0" radius="1"/>
```

Polygon

Wichtig: Ein Polygon muss im Gegenuhrzeigersinn definiert werden und darf nicht mehr als 8 Ecken (vertex) haben.

```
<polygon x="0" y="0">
    <vertex x="0" y="0"/>
    <vertex x="1" y="1"/>
    ...
</polygon>
```

Physik

Bei `<physics>...</physics>` können spezifische Änderungen an den physikalischen Eigenschaften des Bauteils vorgenommen werden. Diese werden im Editor als Werte von Bausatz-Bauteilen verwendet, welche dann somit nicht mehr von der Simulation geerbt werden. (zu den Beschreibungen der Werte siehe Seite 15)

- Dichte: `<density>1</density>`
- Reibung: `<friction>0.5</friction>`
- Elastizität: `<restitution>0.25</restitution>`
- Dämpfung: `<damping linear="0.1" angular="0.25"/>`
- Feste Rotation ein/aus: `<rotation fixed="true/false"/>`
- Feste Position ein/aus: `<position fixed="true/false"/>`

Darstellung

Die Darstellung des Bauteils kann in `visual` angepasst werden:

```
<visual color="#RRGGBB" alpha="1" layer="3"/>
```

Damit kann man die Farbe (color), deren Transparenz (alpha), sowie die Ebene (layer) einstellen. Bauteile mit einer höheren Ebene als andere Bauteile werden immer über den niedrigeren Bauteilen gezeichnet, sodass sie bei einer Überlappung sichtbar sind. Es gibt 10 Ebenen, von Ebene 1 bis Ebene 10.

Ist ein Bild für das Bauteil vorhanden, können folgende Attribute hinzugefügt werden:

```
image="bilder/bild1.png" pixelspermeter="30" x="0" y="0"
```

Als Wert für image ist ein relativer Dateipfad inklusive Dateiname und Dateiendung anzugeben.

Mit 'pixelspermeter' (Pixel pro Meter) kann das Bild skaliert werden: Ist das Bild z.B. 100x50 Pixel gross und das Bauteil 2x1 Meter, sollte hier der Wert 50 eingesetzt werden, damit das Bild genau den physikalischen Bereich des Bauteils abdeckt. Mit x und y wird die linke obere Ecke der Bildes positioniert, im Beispiel sollte also x="-1" und y="0.5" gesetzt werden.

Probleme & Lösungen

Während der Programmierarbeit bin ich auf einige Probleme gestossen. Hier liste ich die wichtigsten auf, je mit Problembeschreibung, Ursache und Lösung.

P: Es gibt seltsame Fehler, welche nicht bei jeder (gleichen) Programmausführung auftreten.

> L: Es greifen zwei oder mehr Threads gleichzeitig auf ein Objekt zu, z.B. wurde in `init()` und in `start()` eines Applets je ein Thread gestartet, welche beide auf das selbe Runnable zugreifen.

P: Components erscheinen nicht dort, wo sie mit `setBounds()` oder `setLocation()` platziert wurden.

> L: Ein übergeordneter Container hat ein Layout (standardmässig FlowLayout), welches Components ausrichtet. Wenn man ein absolutes Layout verwenden will (also alle Components manuell platzieren will), muss man im übergeordneten Container `setLayout(null)` aufrufen.

P: Die Methoden `keyPressed()`, `mouseMoved()`, etc., welche in einem Component mit entsprechenden implementierten Interfaces überschrieben wurden, funktionieren nicht.

> L: Damit Events von einem Listener verarbeitet werden können, müssen sie erst noch einem Component hinzugefügt werden, was im Component mittels `addXyzListener(this)` erreicht werden kann. Diese Anweisung steht normalerweise im Konstruktor.

P: Events werden zweimal ausgeführt, obwohl nur ein Listener registriert ist.

> L: Es ist doch ein zweiter Listener registriert, aber von einer anderen Klasse aus, vielleicht befindet sich in der Superklasse ein `addXyzListener(this)` oder in einem Elterncontainer irgendein `child.addXyzListener(child)`.

P: Ein Component flackert, welcher seine `paint()`-Methode in kurzen Intervallen aufruft.

> L: Wenn ein Component gezeichnet wird, wird zuerst seine Fläche geleert, was meistens bedeutet, dass sie weiss wird. Danach wird er von der `paint()`-Methode gezeichnet. Dauert das Zeichnen zu lange, sieht man die weisse Fläche kurz aufflackern. Um das zu umgehen, wird double buffering (engl. für Doppelpufferung) eingesetzt. Beim double buffering wird nicht direkt auf den Bildschirm gezeichnet, sondern auf ein Bild im Arbeitsspeicher, den Buffer. Nach dem Zeichnen wird dann das Bild auf den Bildschirm kopiert.

P: Java versucht, eine Klasse über das Internet zu laden, und gibt einen `incompatible magic value` Fehler aus.

> L: die Datei wurde nicht gefunden und es wurde eine HTML-Fehlerseite zurückgegeben, bei welcher jedoch der HTTP 404-Header fehlt. Als Lösung macht man einfach seine eigene Fehlerseite im .htaccess, was z.B. so aussehen kann: `ErrorDocument 404 "HTTP Error 404 - Not Found"`.

P: Man will eine Datei mithilfe der Klasse File von einem Webserver herunterladen, bekommt aber eine `java.security.AccessControlException`.

> L: File will standardmässig Schreibrechte, auch wenn die Datei nur zum Lesen geöffnet werden soll. Um das zu umgehen, kann man einfach einen InputStream verwenden. Viele Funktionen, welche ein File als Argument benötigen, gibt es auch mit einem InputStream als Argument. Ist dem nicht so, muss man den InputStream selber auslesen, zum Beispiel mit einem InputStreamReader oder einem BufferedReader.

P: Einige Werte werden zwischen zwei Applet-Starts zwischengespeichert.

> L: Statische Variablen werden im Cache gespeichert und bleiben somit zwischen zwei Starts erhalten. Möchte man das umgehen, müssen die statischen Variablen irgendwo im ausgeführten Code zugewiesen werden, am vorteilhaftesten in der `init()`-Funktion des Applets oder irgendeiner anderen Funktion, welche ebenfalls nur beim Start des Applets ausgeführt wird.

Ausblick

Das Programm kann noch erweitert werden, und zwar sowohl im Hinblick auf die Benutzeroberfläche als auch auf die Unterstützung der Physikengine, welche noch weitere nützliche Einstellungen zulässt. Hier nun eine Liste von Dingen, welche das Programm verbessern könnten:

- Eine lokal ausführbare Variante des Programms
- Bessere Unterstützung der Physikengine:
  - Unterstützung von Joints (Verbindungen zwischen Bauteilen). Dies umfasst unter anderem Federn und Drehpunkte.
  - Bessere Unterstützung von zusammengesetzten Bauteilen (Bauteile, welche mehr als eine Form haben), insbesondere im Bezug auf die physikalischen Eigenschaften der einzelnen Formen
  - Trägheitsmomente der Bauteile einstellen können
  - Physikalische Einstellungen für den Rahmen vornehmen können
- Anzahl Berechnungen pro Sekunde einstellen können
- Mehr Teile in der Standardbibliothek
- Beim Speichern den Namen und die Beschreibung der Simulation angeben können
- Beschriftung der Unterteilungen von Graphen
- Spuren für Bauteile, so dass man sieht, wo ein Bauteil bereits gewesen ist.

Danksagung

Ich möchte einigen Personen, welche mir bei der Arbeit geholfen haben, danken:

Meiner Schwester für ihre Unterstützung und Ideen

Meinen Eltern für ihre Anregungen und das Korrekturlesen

Meinem Betreuer Rainer Steiger für die Hilfe bei der schriftlichen Arbeit

Einigen Benutzern vom Java-Forum[9] für die Hilfe bei einigen kniffligen Problemen

9 http://www.java-forum.org/

Anhang

Stichwortverzeichnis

Quellenverzeichnis

http://www.playcrafter.com/ – Idee, unter anderem das Bibliothek/Bausatz-Konzept

http://www.jbox2d.org/ – Physikengine

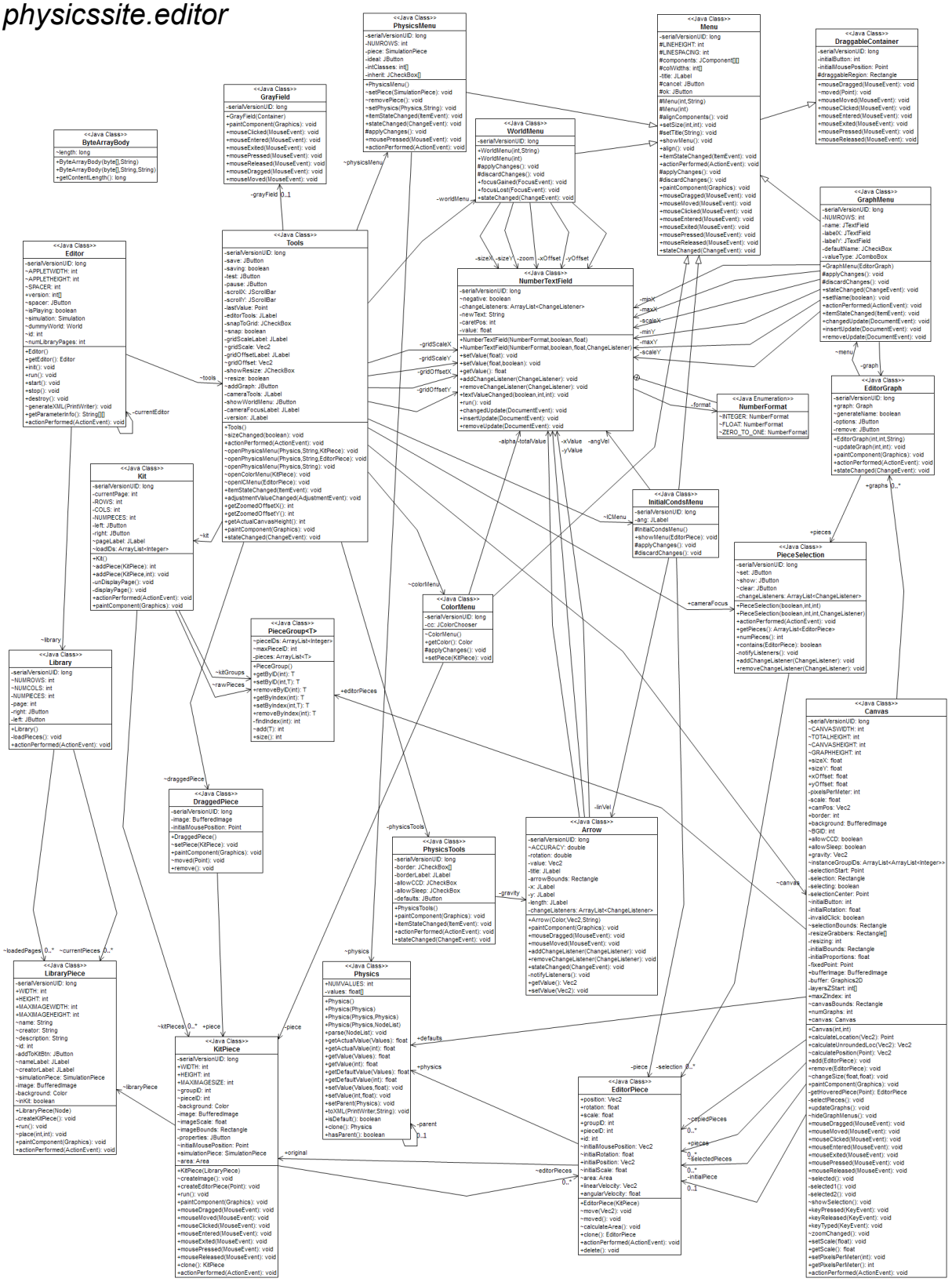
http://hc.apache.org/ – Apache HttpComponents für das Speichern einer Simulation via HTTP

http://radomirml.com/2009/02/13/file-upload-with-httpcomponents-successor-of-commons-httpclient – Bugfix für HttpComponents

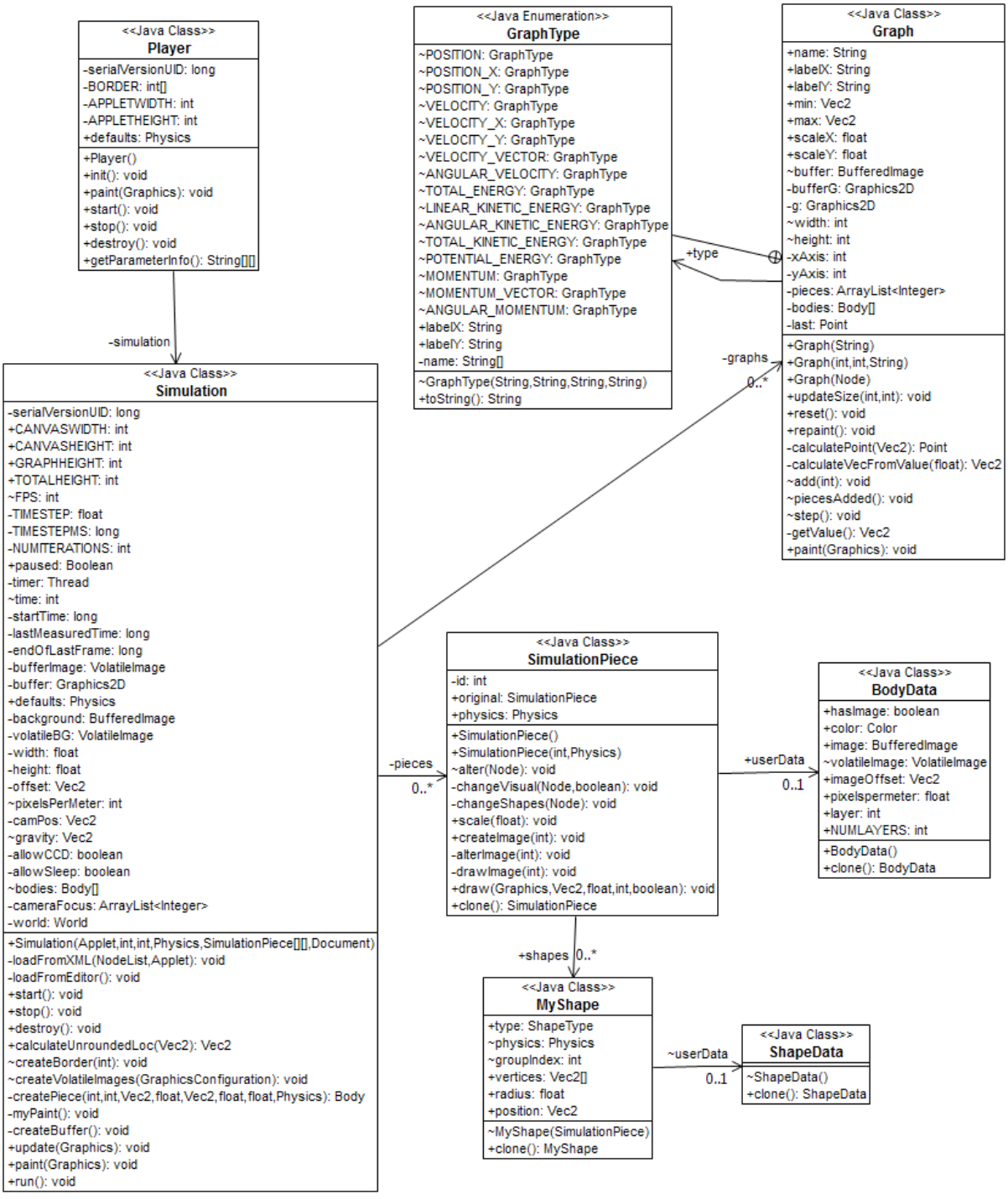
http://www.petrastumpf.de/michael/Wissen/SpieleprogrammierungInJava.pdf – nützliche Informationen über das Rendern in Java

Klassendiagramme

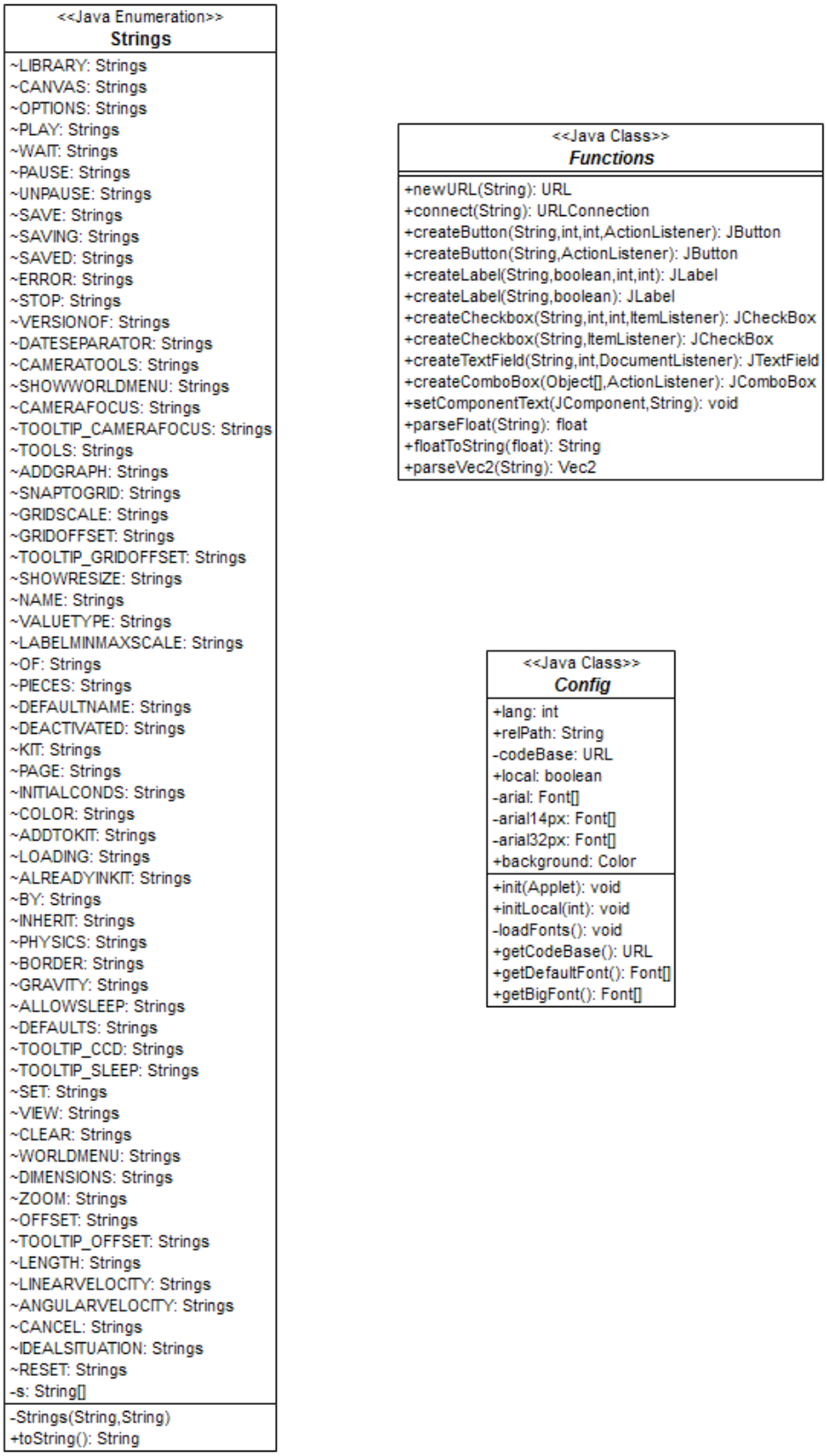
physicssite.editor



physicssite.player



physicssite.common



Hierarchiediagramm