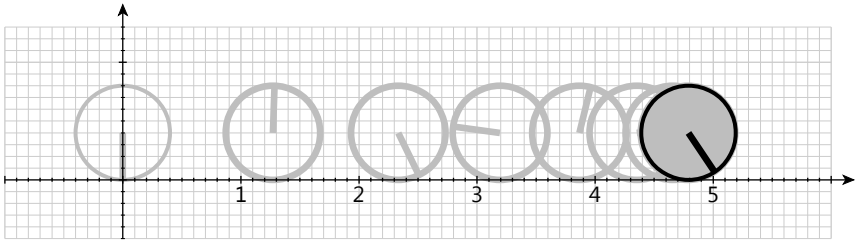
Aufgabenblatt 5

Rollender Ball



In diesem Aufgabenblatt soll ein Ball simuliert werden, der auf einer horizontalen Unterlage rollt. Es seien $r = 0{,}4\,m$ der Durchmesser des Balls und $m = 5\,kg$ dessen Masse. Der Ball bewege sich entlang der x-Achse. Die Variable *x* beschreibt die Position des Balls entlang der x-Achse und die Variable *v* dessen Geschwindigkeit. Zu Beginn der Simulation, also zum Zeitpunkt $t = 0$, habe *x* den Wert 0 und *v* den Wert $2{,}3\,\frac{m}{s}$.

Der Ball erfährt während der Rollbewegung einen Rollwiderstand. Der Rollwiderstandskraft $F_r = -m \cdot g \cdot c_r$ ist entgegen der Bewegung gerichtet und bremst den Ball ab. Dabei ist $g = 9{,}81\,\frac{m}{s^2}$ die Erdbeschleunigung und c_r der Rollwiderstandsbeiwert. Der Rollwiderstandsbeiwert hängt vom Material Balles und vom Material der Unterlage ab. In diesem Experiment sei $c_r = 0{,}05$. Es sei a die Beschleunigung, die auf den Ball wirkt. Es gilt: $a = \frac{F_r}{m}$.

Die Rollwiderstandskraft wirkt genau so lange wie sich der Ball bewegt. Sobald der Ball zum Stehen gekommen ist, wirkt diese Kraft nicht mehr. In dem Augenblick, in dem der Ball zum Stehen kommt, springt der Wert der Variable F_r abrupt von $-m \cdot g \cdot c_r$ auf 0.

1. Teilaufgabe

Programmieren Sie ein physikalisches System mit dem Namen *RollenderBall*. Fügen Sie in die Klasse *RoolenderBall* alle relevanten physikalischen Variablen ein. Versehen Sie die Variablen mit den richtigen Anfangswerten, physikalischen Einheiten und Ableitungsbeziehungen. Vereinfachend soll zunächst angenommen werden, dass F_r während der gesamten Simulation konstant den Wert $F_r = -m \cdot g \cdot c_r$ hat.

Die relevanten physikalischen Variablen sollen im Plotter dargestellt werden. Ergänzen Sie das Programm entsprechend!

Starten Sie die Simulation!

2. Teilaufgabe

Zu dem physikalischen System soll ein Grafikkomponente mit dem Namen *RollenderBallTVG* programmiert werden. Der folgende Programmcode zeigt das Grundgerüst für die Klasse *RollenderBallTVG*. Übernehmen Sie diesen Programmcode.

```
import de.physolator.usr.tvg.TVG;
import de.physolator.usr.tvg.Shape;

public class RollenderBallTVG extends TVG {

    private RollenderBall rb;

    public RollenderBallTVG(RollenderBall1 rollenderBall) {
        rb = rollenderBall;
        geometry.setUserArea(-1, 6, -0.5, 1.3);
        geometry.setRim(30, 30, 30, 30);
        scalesStyle.visible = true;
    }

    public void paint() {
        style.useUCS = true;
        style.strokeWidth = 3;
        // Platz für Zeichenbefehle
    }
}
```

Die Grafikkomponente *RollenderBallTVG* soll angezeigt werden, sobald das physikalische System *RollenderBall* geladen wird. Ergänzen Sie dazu den Programmcode von *RollenderBall* um die folgenden Zeilen:

```
    public void initGraphicsComponents(GraphicsComponents g) {
        g.addTVG(new RollenderBall1TVG(this));
    }
```

Mit dieser Ergänzung und nach dem erneuten Laden des physikalische Systems *RollenderBall* (reload) erscheint im Physolator eine Grafikkomponente. In der Grafikkomponente erkennt man bereits die beiden Koordinatenachsen und die Gitternetzlinien. Der Zeichenbereich reicht in x-Richtung von -1 bis 6 und in y-Richtung von -0.5 bis 1,3.

In diese Grafikkomponente sollen in der nachfolgenden Teilaufgabe der aktuelle Zustand des physikalischen Systems eingezeichnet werden. Dazu sollen an der markierten Stelle in der paint-Methode entsprechende Zeichenbefehle eingefügt werden. Innerhalb der Grafikkomponente *RollenderBallTVG* kann auf das physikalische System *RollenderBall* zugegriffen werden. Die Variable, die das physikalische System, enthält hat den Namen *rb* und ist vom Typ *RollenderBall*. Möchte man in der *paint*-Methode auf die aktuelle Position des Balles zugreifen, so findet man diese unter *rb.x*, die aktuelle Geschwindigkeit findet man unter *rb.v*, etc..

Erläuterungen zum Programmcode

Anders als bei den Grafiken aus den bisherigen Aufgabenblättern, soll in dieser Grafik der aktuelle Zustand eines physikalischen Systems dargestellt werden. Damit die Grafikkomponente Zugriff auf das physikalische System bekommt, wird beim Konstruktoraufruf der Grafikkomponente

new RollenderBall1TVG(this)

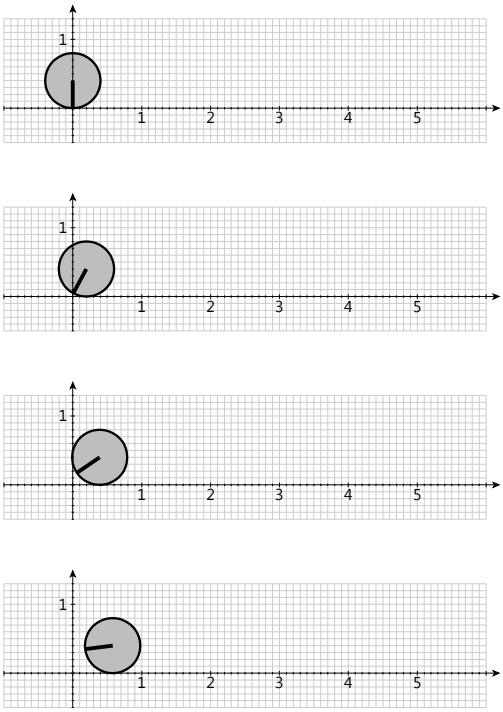
dem Konstruktor das physikalische System übergeben. Dabei steht *this* für das eigene Objekt, also die aktuelle Instanz von *RollenderBall*, also das aktuelle physikalische System. Der Konstruktor von *RollenderBallTVG* speichert diese mit

rb = rollenderBall;

im Objektattribut *rb* ab. Aus der Methode *paint* kann dann nachfolgend auf dieses Objekt zugegriffen werden.

3. Teilaufgabe

Der Ball soll jetzt in die Grafik an der aktuellen Position *rb.x* eingezeichnet werden. Die Unterlage, auf der der Ball rollt, sei die x-Achse. Der Mittelpunkt des Balles befindet sich somit an der Position *(rb.x, rb.r).*



Der Ball soll durch einen Kreis dargestellt werden. Zur Besserung Veranschaulichung der Rollbewegung soll der Ball eine Linie als Markierung erhalten. Die Linie reicht von der Mitte des Balles bis zur Kreislinie und zeigt zu Beginn nach unten. Durch die Rollbewegung des Balles dreht sich diese Linie im Uhrzeigersinn. Hinweis: Die Richtung der Markierung hängt nur von der aktuellen Position des Balles ab. Bestimmen Sie zunächst den Drehwinkel der Markierung in Abhängigkeit von der aktuellen Position des Balles.

Zeichnen Sie den Ball inklusive seiner Markierung in die Grafik ein. Fügen Sie dazu geeignete Zeichenbefehle in die Methode *paint* ein.

drawLine(x1,y1,x2,y2);	zeichnet eine Linie von *(x1,y1)* nach *(x2,y2)*
drawCircle(x,y,r, Shape.POLYGON_LINE_LOOP);	zeichnet einen gefüllten Kreis mit Mittelpunkt *(x,y)* und Radius *r*

Zeichnen Sie die Markierungslinie mit einer Pixelbreite von 5, damit diese Linie besser erkannt werden kann. Fügen Sie dazu vor dem Zeichenbefehl für die Markierung folgende Zuweisung ein:

```
style.strokeWidth = 5;
```

4. Teilaufgabe

Es soll herausgefunden werden, wie lange der Ball rollt und wie weit er kommt.

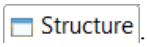
Bisher erfährt der Ball während der gesamten Simulation eine Beschleunigung von $F_r = -m \cdot g \cdot c_r$. Die Variable F_r erhält diesen Wert zu Beginn der Simulation und während der Simulation wird er bisher nicht verändert. Jetzt soll der Wert von F_r genau in dem Zeitpunkt auf *0* gesetzt werden, an dem *v* die

Geschwindigkeit *0* erreicht. Fügen Sie dazu das folgende Stück Programmcode in die Klasse *RollenderBall* ein.

```
public void g(double t, AfterEventDescription afterEventDescription) {
    if (v < 0)
        afterEventDescription.reportEvent(() -> {
            Fr = 0;
        });
}
```

Diese Ergänzung im Programmcode bewirkt, dass bei der Simulation der Zeitpunkt bestimmt wird, zu dem *v* den Wert *0* unterschreitet. Der Physolator berechnet diesen Zeitpunkt durch Intervallschachtelung und er berechnet den Zustand zu diesem Zeitpunkt. Sobald das physikalische System diesen Zustand erreicht hat, wird die Zuweisung *Fr=0;* ausgeführt. Ein solcher Zeitpunkt wird als ein physikalisches Ereignis bezeichnet.

Gehen Sie mit der Maus in den Plotter. Es erscheint der Button für die Einstellungen  . Drücken Sie diesen Button. Es erscheint ein Dialog mit den TVG-Parametern. Gehen Sie dort zu „plot“ und kreuzen Sie „show time line“ an. Unterhalb des Plot-Bereichs erscheint eine zusätzliche Zeitachse. Führen Sie nun die Simulation in Einzelschritten durch, indem Sie wiederholt auf den Button  drücken. Sie erkennen, dass die Simulation in Schritten von 0,05s voranschreitet. Jeder Zustand wird auf der Zeitachse durch einen kleinen grauen Strich dargestellt. Zu dem Zeitpunkt des physikalischen Ereignisses wird außer der Reihe ein zusätzlicher Zeitpunkt und der Zustand des physikalischen Systems zu diesem Zeitpunkt berechnet. Auf der Zeitachse sind die zusätzlichen Zeitpunkte der physikalischen Ereignisse durch etwas größere schwarze Striche dargestellt.

Gehen Sie dazu Schritt für Schritt bis zu dem Zeitpunkt des physikalischen Ereignisses. Lesen Sie den Zeitpunkt ab, zu dem das Ereignis eingetreten ist und den *x*-Wert zu diesem Zeitpunkt. Hinweis: Die Variablenwerte finden Sie im Bereich „Structure“ 

.

Der Physolator bestimmt die Zeitpunkte von physikalischen Ereignissen durch Intervallschachtelung. Die Genauigkeit der Zeitwerte beträgt 12 Dezimalstellen. Das ist zwar sehr genau, aber nicht exakt. Aus diesem Grund ist der Wert von *v* zu diesem Zeitpunkt auch nicht exakt 0, sondern weicht geringfügig von 0 ab. Um zu erzwingen, dass sich der Ball nach dem physikalischen Ereignis überhaupt nicht mehr bewegt – auch nicht geringfügig – soll der Wert von *v* zu diesem Zeitpunkt auch auf 0 gesetzt werden. Ändern Sie Ihren Programmcode wie folgt ab und starten Sie die Simulation erneut.

```
public void g(double t, AfterEventDescription afterEventDescription) {
    if (v < 0)
        afterEventDescription.reportEvent(() -> {
            Fr = 0;
            v = 0;
        });
}
```

5. Teilaufgabe

Der Strömungswiderstand (Luftwiederstand) wurde bisher nicht berücksichtigt. Jetzt soll erneut die Frage beantwortet werden, wie lange der Ball rollt und wie weit er dabei kommt – dieses Mal sollen aber sowohl der Rollwiderstand als auch der Strömungswiderstand berücksichtigt werden.

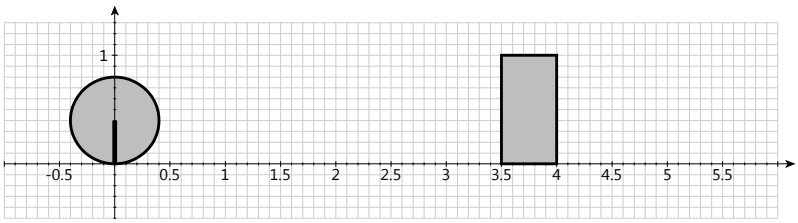
Erweitern Sie dazu das bestehende physikalische System derart, dass bei der Berechnung der Kraft neben der Rollreibung auch der Strömungswiderstand berücksichtigt wird. Für den Strömungswiderstand F_L gilt nachfolgende Gleichung. Beachten Sie dabei, dass der Strömungswiderstand F_L immer der Bewegungsrichtung *v* entgegen gerichtet ist.

$$F_L = \frac{1}{2} A c_w \rho v^2$$

In dieser Gleichung ist $A = r^2 \pi$ die Querschnittsfläche der Kugel, $c_w = 0{,}4$ der Strömungswiderstandskoeffizient für einen kugelförmigen Körper und $\rho = 1{,}2041 \frac{kg}{m^3}$ die Dichte der Luft.

Hinweis: Die Formel $F_L = \frac{1}{2} A c_w \rho v^2$ liefert immer einen positiven Wert. Damit die Kraft immer der Bewegungsrichtung *v* entgegensteht, multipliziert man diesen Wert mit *-sgn(v)*. Die Signumfunktion *sgn* bildet positive Zahlen auf 1 ab, negative Zahlen auf -1 und die 0 auf 0. In der Programmiersprache Java heißt diese Funktion *signum* und ist Teil des Pakets *Math*.

6. Teilaufgabe



Hinzu kommt jetzt eine Mauer an der Position $x_w = 3{,}5\,m$. Ergänzen Sie die Grafikkomponente *RollenderBallTVG* so, dass diese Wand gemäß obiger Abbildung eingezeichnet wird.

drawRectangle(x1,y1,x2,y2,Shape.POLYGON_LINE_LOOP);	zeichnet ein Achsen-paralleles Rechteck, *(x1,y1)* ist die linke untere Ecke, *(x2,y2)* die Ecke rechts oben

Beim Prall gegen die Wand erfährt der Ball einen Stoß mit der Stoßzahl $k = 0{,}7$. Beim Stoß gegen die Wand ändert der Ball abrupt seine Richtung und rollt dann in die entgegengesetzte Richtung. Durch den Prall verliert der Ball an Geschwindigkeit. Ist die Geschwindigkeit unmittelbar vor dem Stoß *v*, so ist die Geschwindigkeit unmittelbar nach dem Stoß $-k\,v$.

Erneut soll berechnet werde, wie lange der Ball rollt und an welcher Position er zum Stehen kommt. Ergänzen Sie dazu den Programmcode entsprechend.

Hinweise

Die Methode *g* enthält bereits ein *if*-Konstrukt, das das physikalische Ereignis des Anhaltens realisiert. Fügen Sie in analoger Weise ein weiteres *if*-Konstrukt hinzu, das den Stoß des Balles gegen die Wand realisiert.

Anders als bei der vorangegangenen Teilaufgabe gibt es in diesem Szenario zwei Bewegungsrichtungen: vor dem Stoß bewegt sich der Ball nach rechts, anschließend nach links. Ob der Ball die Schwelle zum Stillstand überschreitet, wurde bisher im Programmcode mit *v<0* überprüft. So einfach ist das nun nicht mehr. Vor dem Stoß bewegt sich der Ball nach rechts und es muss geprüft werden, ob *v<0* ist, und nach dem Stoß bewegt sich der Ball nach links und es muss geprüft werden, ob *v>0* ist.

Empfehlung: Führen Sie eine boolesche Variable mit dem Namen *rechtsbewegung* ein. Fügen Sie dazu die folgende Zeile in den Programmcode der Klasse *RollenderBall* ein.

```
public boolean rechtsbewegung = true;
```

Zu Beginn erhält die Variable den Wert *true.* Durch den Stoß soll nicht nur *v* zu $-k\,v$ verändert werden, sondern es soll auch die Variable *rechtsbewegung* auf *false* gesetzt werden. Auf diesen booleschen Wert kann zugegriffen werden, um zusammen mit dem Wert *v* zu entscheiden, ob der Ball die Schwelle zum Stillstand überschreitet.