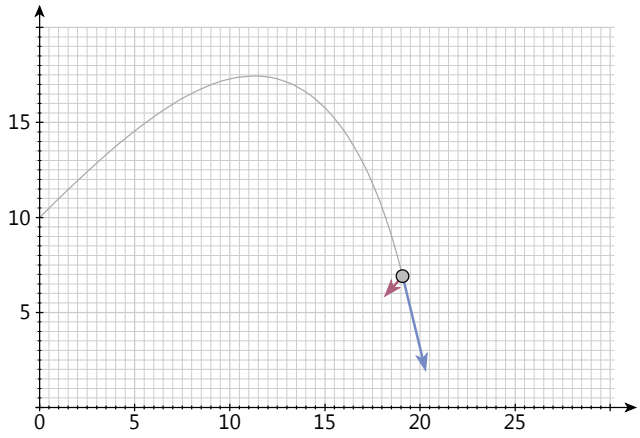
Aufgabenblatt 6

Schiefer Wurf



In diesem Aufgabenblatt soll die Wurfbahn eines Balles berechnet werden. Während des Fluges erfährt der Ball durch die Luft eine Strömungswiderstandskraft, die der Bewegung entgegen gerichtet ist. Die Flugbahn ist abhängig von verschiedenen Parametern wie der Anfangsposition, der Anfangsgeschwindigkeit, dem Durchmesser des Balles und der Masse. In Abhängigkeit von diesen Parametern soll bestimmt werden, wie weit der Ball fliegt und wie hoch er während des Fluges kommt.

Die Flugbahn des Balles, seine aktuelle Geschwindigkeit und die aktuelle Beschleunigung sollen grafisch dargestellt werden. Analog zu Aufgabenblatt 5 könnte man eine solche Grafikkomponente selbst programmieren. Stattdessen soll in diesem Aufgabenblatt eine bereits bestehende Grafikkomponente mit dem Namen *MechanicsTVG* eingesetzt werden. Diese Grafikkomponente dient der Visualisierung von physikalischen Systemen, die aus punktförmigen Massen bestehen. Mit *MechanicsTVG* können für ein oder mehrere punktförmige Massen deren Flugbahn, deren Geschwindigkeit und deren Beschlenigung grafisch dargestellt werden. *MechanicsTVG* kann in beliebigen physikalischen Systemen eingesetzt werden. *MechanicsTVG* enthält verschiedene Parameter, mit deren Hilfe *MechanicsTVG* an die unterschiedlichen Anforderungen der physikalischen Systeme angepasst werden kann.

Die Grafikkomponente *MechanicsTVG* wird nicht nur in diesem Aufgabenblatt, sondern auch in mehreren weiteren Aufgabenblättern zum Einsatz kommen.

Vorbereitung

Binden Sie zunächst die Library *mechanics.jar* in Ihr Projekt ein. Die Datei *mechanics.jar* finden Sie auf der Webseite im Bereich dieser Aufgabe. Kopieren Sie diese Datei in Ihr Projektverzeichnis. Fügen Sie diese Library zu Ihrem Classpath hinzu. In Eclipse klicken Sie dazu die Datei *mechanics.jar* an – rechte Maustaste – build path – add to build path. Fertig. Ab sofort können Sie in Ihrem Projekt die Klassen in dieser Library verwenden, insbesondere die Klasse *MechanicsTVG*.

Übernehmen Sie den folgenden Programmcode:

```
import static java.lang.Math.*;
import mechanics.tvg.MechanicsTVG;
import de.physolator.usr.*;

public class SchieferWurf extends PhysicalSystem {

    public void initGraphicsComponents(GraphicsComponents g, Structure s, Recorder r,
                    SimulationParameters sp) {
        MechanicsTVG t = new MechanicsTVG(this, s, r);
        t.geometry.setUserArea(0, 150, 0, 50);
        t.showPaths = true;
        t.showVelocity = true;
        t.showAcceleration = true;
        t.velocityScaling = 1;
        t.accelerationScaling = 1;
        t.showLabels = false;
        t.addPointMass("x", "y", "vx", "vy", "ax", "ay");
        g.addTVG(t);
    }

}
```

Erläuterungen zum Programmcode

Dieser Programmcode verwendet die Grafikkomponente *MechanicsTVG*. Dazu wird in der Methode *initGraphicsComponents* zunächst eine Instanz dieser Klasse erzeugt und es werden nachfolgend mehrere Einstellungen vorgenommen, mit deren Hilfe die Grafikkomponente an die Erfordernisse des physikalischen Systems *SchieferWurf* angepasst wird. Der sichtbare Bereich des Koordinatensystems festgelegt wird – wie schon in Aufgabenblatt 5 – mit *setUserArea* festgelegt. Es wird festgelegt, dass die Pfade der punktförmigen Massen angezeigt werden sollen (showLabels) und dass deren Geschwindigkeit und durch Beschleunigung durch Pfeile dargestellt werden sollen (showVelocity, showAcceleration). Mit *velocityScaling* und *accelerationScaling* wird schließlich der Zusammenhang zwischen Vektorlänge und Geschwindigkeit bzw. Beschleunigung festgelegt.

In diesem Aufgabenblatt soll nur ein punktförmige Masse dargestellt werden, nämlich der Ball. Die Position des Balles sei *(x,y)*, seine Geschwindigkeit *(vx,vy)* und seine Beschleunigung *(ax,ay)*. Die Methode *addPointMass* legt fest, dass die Grafikkomponente *MechanicsTVG* in dem physikalischen Systeme auf die Variablen *x, y, vx, vy, ax* und *ay* zugreifen soll, um diese punktförmige Masse grafisch darzustellen. Das physikalische System *SchieferWurf* enthält diese Variablen noch nicht, sie sollen jedoch in der ersten Teilaufgabe hinzugefügt werden.

Mit *MechanicsTVG* können prinzipiell mehrere punktförmige Massen dargestellt werden. In diesem Fall kann und sollte man die einzelnen punktförmigen Massen beschriften. In diesem Aufgabenblatt soll darauf verzichtet werden. Deshalb wird in obigem Programmcode *showLabels* der Wert *false* zugewiesen.

1. Teilaufgabe

Ein Ball wird geworfen. Die Anfangsposition des Balles sei $\begin{pmatrix} 0 \\ 10 \end{pmatrix} m$, seine Anfangsgeschwindigkeit $\begin{pmatrix} 20 \\ 20 \end{pmatrix} \frac{m}{s}$. Der Ball erfährt eine Erdbeschleunigung, die ihn mit $g = 9{,}81 \frac{m}{s^2}$ nach unten zieht.

Programmieren Sie dieses physikalische System analog zu Aufgabenblatt 1. Ergänzen Sie dazu den vorgegebenen Programmcode um Deklarationen für die physikalische Variablen, legen Sie die Anfangswerte der physikalischen Variablen fest und definieren Sie die Ableitungsbeziehungen zwischen den physikalischen Variablen. Falls benötigt: Programmieren Sie eine Methode f mit den Formeln zur Bestimmung der abhängigen Variablenwerte.

Verwenden Sie die Variablennamen *x, y, vx, vy, ax* und *ay*, wobei *(x,y)* für die Position des Balles, *(vx,vy)* für dessen Geschwindigkeit und *(ax,ay)* für die Beschleunigung steht, die der Ball erfährt. Nach dem Laden und dem Starten des physikalischen Systems erkennt man die Grafikkomponente *MechanicsTVG*, die den Ablauf des Wurfes auf dem Bildschirm darstellt.

2. Teilaufgabe

Der Strömungswiderstand wurde bisher nicht berücksichtigt. Erweitern Sie das bestehende physikalische System derart, dass bei der Berechnung der Beschleunigung (Teilaufgabe 1) auch der Strömungswiderstand berücksichtigt wird. Man nehme an, der Ball habe eine Masse von $m = 0{,}1\,kg$ und einen Radius von $r = 0{,}1\,m$. 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 des Balles, $c_w = 0{,}4$ der Strömungswiderstandskoeffizient für einen kugelförmigen Körper und $\rho = 1{,}2041 \frac{kg}{m^3}$ die Dichte der Luft.

Durch den Strömungswiderstand fliegt der Ball deutlich weniger weit. Passen Sie die Parameterwert von *setUserArea* in geeigneter Weise an.

3. Teilaufgabe

Es soll bestimmt werden, wie hoch der Ball fliegt. Außerdem soll bestimmt werden, wo sich dieser Punkt der größten Höhe befindet und zu welchem Zeitpunkt er diesen Punkt erreicht.

Der Zeitpunkt, in dem der Ball den höchsten Punkt erreicht, ist dadurch gekennzeichnet, dass zu diesem Zeitpunkt *vy* den Wert 0 hat. Zu Beginn ist *vy* positiv. In dem Augenblick, in dem der Ball den höchsten Punkt erreicht, unterschreitet *vy* den Wert 0. Fügen Sie den folgenden Programmcode in Ihre Klasse ein:

```
public ThresholdTrigger tr1 = new ThresholdTrigger(() -> vy < 0);
```

Dieser zusätzliche Programmcode bewirkt, dass immer dann, wenn *vy* den Grenzwert 0 über- oder unterschreitet ein Event ausgelöst wird. Die zentrale Information in diesem Programmstück ist die Bedingung *vy<0*. Der Wahrheitswert dieser Bedingung wechselt beim Über- beziehungsweise Unterschreiten zwischen *true* und *false*.

Der Name der Variable *tr1* hat keine besondere Bedeutung. Arbeitet man in einem physikalischen System mit mehreren Triggern, so muss man jeder Variablen einen anderen Namen geben. Man kann die Trigger dann beispielsweise mit *tr1, tr2, tr3,...* bezeichnen.

Ersetzen Sie nun den bisherigen Trigger durch den folgenden Code:

```
public ThresholdTrigger tr1 = new ThresholdTrigger(() -> vy < 0)
                .setName("Hochpunkt").setInfo(() ->  "x="+x + " y="+y).setDoPrint(true);
```

Anders als beim bisherigen Programmcode, ordnet dieser Programmcode dem Ereignis einen Namen zu, nämlich „Hochpunkt“. Diese Zusatzinformation wird relevant, wenn man mit verschiedenen Ereignssen arbeitet. Durch den Methodenaufruf *setDoPrint(true)* wird festgelegt, dass zum Zeitpunkt des Events eine Ausgabe auf der Konsole erfolgen soll. Auf der Konsole erscheint dann eine einzeilige Meldung, die besagt, dass ein Schwellwert-Ereignis eingetreten ist, dazu der Namen (falls vorhanden) und der Zeitpunkt. Mit der Methode *setInfo* können in dieser Zeile neben der Zeit noch andere Variablenwerte ausgegeben werden. In diesem Fall werden die Werte von *x* und *y* ausgegeben.

4. Teilaufgabe

Bestimmen Sie, wie weit der Ball fliegt. Bestimmen Sie analog zu Teilaufgabe 3 wann und wo der Ball den Boden (*y=0*) erreicht.

5. Teilaufgabe

Es soll bestimmt werden, wo sich der zum Zeitpunkt *t=0,27134s* aufhält.

Der Physolator geht bei der Simulation Schritt für Schritt vor und berechnet zu jedem Zeitpunkt die Variablenwerte. Die Schrittweite kann variiert werden. Je nach gewählter Einstellung ist es auch möglich, dass der Physolator die Schrittweite für jeden Schritt selbst bestimmt und sich bei der Bestimmung der Schrittweite an bestimmten Genauigkeitszielen orientiert. Man kann deshalb nicht davon ausgehen, dass der Physolator, den Zustand des physikalischen Systems zum Zeitpunkt *t=0,27134s* bestimmt. Um zu erzwingen, dass der Zustand zum Zeitpunkt *t=0,27134s* bestimmt wird, muss man zu diesem Zeitpunkt ein physikalisches Ereignis erzeugen. Dafür kann der *TimeTrigger* verwendet werden.

Fügen Sie den folgenden Programmcode in Ihre Klasse SchieferWurf ein:

```
public TimeTrigger tr3 = new TimeTrigger(0.27134)
                    .setName("Zeitpunkt 1").setInfo(() -> "x=" + x + " y=" + y).setDoPrint(true);
```

Die Funktionsweise der Klasse *TimeTrigger* ähnelt der der Klasse *ThresholdTrigger*. Statt einer Schwellwertbedingung wird dem Konstruktor ein Zeitpunkt übergeben. Auch bei dieser Klasse gibt es in analoger Weise die Methoden *setName*, *setInfo* und *setDoPrint.*

Im nächsten Schritt soll die Position des Balles zu mehreren Zeitpunkten bestimmt werden: *0,27134s, 0,4525s* und *1.273s.* Verwenden Sie dafür den folgenden Programmcode anstelle des bisherigen Time-Triggers *tr3*. Auf einen besonderen Namen für die Events wird in diesem Programmcode verzichtet.

```
public TimeTrigger tr4 = new TimeTrigger(0.27134, 0.4525, 1.273)
                    .setInfo(() -> "x=" + x + " y=" + y).setDoPrint(true);
```

Als Nächstes soll die Position zu mehreren Zeitpunkten bestimmt werden, die durch eine Folge von Zeitpunkten beschrieben werden. Die Position soll in gleichen zeitlichen Abständen von jeweils *0,27134s* bestimmt werden. Verwenden Sie dazu den folgenden Programmcode.

```
public TimeTrigger tr5 = new TimeTrigger((i)-> (i+1)*0.2713)
                    .setInfo(() -> "x=" + x + " y=" + y).setDoPrint(true);
```

Hinweis: Der erste Zeitpunkt der Zeitpunktfolge soll *0,27134s* ein. Die Zählung bei dieser Zahlenfolge beginnt bei *i=0.* Deshalb ergibt sich der *i*-te Zeitpunkt zu *(i+1)*0.2713*.

6. Teilaufgabe

In dieser Teilaufgabe soll untersucht werden, unter welchem Abwurfwinkel der Ball die größte Höhe beziehungsweise die größte Flugweite erreicht. Der Betrag der Anfangsgeschwindigkeit sei $v_0 = 28\frac{m}{s}$. Die Anfangsposition des Balles sei wieder $\begin{pmatrix} 0 \\ 10 \end{pmatrix} m$. Die beiden Abwurfwinkel für die größte Höhe und die größte Weite sollen mit einer Genauigkeit von 3 Dezimalstellen bestimmt werden. Außerdem soll die größtmögliche Höhe und die größtmögliche Weite bestimmt werden

Empfehlungen: Ergänzen Sie zunächst Ihren Programmcode um die Konstanten *v0* und *phi*. *v0* stehe für die Anfangsgeschwindigkeit, *phi* für den Abwurfwinkel gemessen in Grad gegenüber der Horizontalen. Berechnen Sie aus *v0* und *phi* die Anfangswerte von *vx* und *vy.* Führen Sie mehrere Simulationsläufe durch, beobachten Sie dabei jeweils die erreichte Höhe und Weite bestimmen Sie per Intervallschachtelung den Wert von *phi*, für den den höchste Höhe erreicht wird, und den Wert von *phi*, für den die größte Weite erreicht wird.

7. Teilaufgabe

Jetzt soll wieder, wie in Teilaufgabe 1, ein Wurf ohne den Strömungswiderstand der Luft betrachtet werden – mit der in Teilaufgabe 1 vorgegebene Werten für die Anfangsposition und Anfangsgeschwindigkeit. Bei einem Wurf ohne Strömungswiderstand beschreibt die Bahn des Balles eine Parabel. In diesem Fall können die Wurfhöhe und die Wurfweite auch direkt mit Papier und Bleistift bestimmt werden – ohne Simulation. Bestimmen Sie diese Werte zunächst mit Papier und Bleistift.

Führen Sie anschließend einen Simulationslauf durch. Verwenden Sie dazu den bisherigen Programmcode und setzen Sie die Anfangsgeschwindigkeit wieder auf $\begin{pmatrix} 20 \\ 20 \end{pmatrix} \frac{m}{s}$. Wenn Sie die Luftdichte ρ auf 0 setzen, dann verschwindet der Strömungswiderstand. Vergleichen Sie Ihr selbst errechneten Ergebnisse für die Wurfhöhe und Wurfweite mit dem Simulationsergebnis.