

Kollision zweier Kugeln (2D)

Wechseln zu: [Navigation](#), [Suche](#)

Dieser Artikel wird derzeit von einem Autor gründlich bearbeitet. Die Inhalte sind daher evtl. noch inkonsistent.

Dieser Artikel erfüllt die [GlossarWiki-Qualitätsanforderungen](#) **nur teilweise**:

Korrektheit: 1
(nur rudimentär
überprüft)

Umfang: 1
(zu gering)

Quellenangaben:
1
(fehlen großteils)

Quellenarten: 3
(gut)

Konformität: 5
(ausgezeichnet)

Inhaltsverzeichnis

- [1 Grundannahmen](#)
- [2 A-posteriori-Kollisionserkennung von zwei Kugeln](#)
- [3 A-priori-Kollisionserkennung von zwei Kugeln](#)
- [4 Quellen](#)

1 Grundannahmen

Es sei ein zweidimensionales Ball-Objekt **b** gegeben, das folgende Attribute habe:

x, **y**: Position (Einheit: Pixel)

vx, **vy**: Geschwindigkeit in x - und y -Richtung (Einheit: Pixel/s)

r: Radius (Einheit: Pixel)

m: Masse (die Einheit ist irrelevant, sie muss nur für alle Bälle dieselbe sein)

Die **Frequenz** der Neuberechnung der Positionen aller im System vorhandener Bälle sei **f**, d. h., die Positionen werden **f** mal pro Sekunde Neuberechnet. Wenn es zu keiner Kollision kommt, wird der Ball um vx/f bzw. vy/f Pixel in x - bzw. y -Richtung verschoben:

```
b.x += b.vx/f;  
b.y += b.vy/f;
```

Je höher die Frequenz **f** ist, desto genauer wird die Bewegungsbahn der Bälle simuliert.

Im Folgenden seien $s_1 = (s_{1x}, s_{1y})$ und $s_2 = (s_{2x}, s_{2y})$ die Startpositionen zweier Bälle und $v_1 = (v_{1x}, v_{1y})$ und $v_2 = (v_{2x}, v_{2y})$ die zugehörigen Geschwindigkeitsvektoren.

Es gilt also, dass sich die beiden Bälle zu Zeitpunkt t_0 an den Positionen s_1 und s_2 befinden. Eine Sekunde später, d. h. zum Zeitpunkt t_0+1 befinden sich sie an den Positionen s_1+v_1 und s_2+v_2 , sofern sie nicht kollidieren. Wenn die Neuberechnung mit einer **Frequenz**

f Hz erfolgt, d. h., wenn die **Periodendauer** $t = 1/f$ beträgt, dann befinden sich die Bälle nach der Berechnung an den Positionen $s_1 + tv_1$ und $s_2 + tv_2$, sofern sie nicht kollidieren.

In Folgenden wird stets der Fall $t=1$ angenommen. Alle Aussagen, die für einen *beliebigen* Geschwindigkeitsvektor v gelten, gelten dann natürlich auch für $v' := sv$, wobei s ein beliebiger skalarer Wert sei.

2 A-posteriori-Kollisionserkennung von zwei Kugeln

Beispiel: [WK_Ball02: index40c.html](#)

```
const EPSILON = Number.EPSILON;

function collisionCircleCircle(p_c1, p_c2)
{ // l_n: Normalenvektor (Verbindung zwischen den beiden Kugeln)
  let l_nx = p_c2.x - p_c1.x;
  let l_ny = p_c2.y - p_c1.y;
  // Abstand der beiden Kugeln
  let l_dist = Math.sqrt(l_nx * l_nx + l_ny * l_ny);
  // Die Kugeln dürfen sich nicht an derselben Stelle befinden,
  // weil sie sonst nicht entlang der nicht-existenten Normalen
  // auseinandergesogen werden können.
  // Dieser Fall sollte nur sehr selten eintreten (z.B. wenn eine
  // neue Kugel genau an der Stelle einer existenten Kugel erstellt
  // wird).
  if (l_dist < EPSILON)
  { p_c2.x = p_c2.x + p_c2.r;
    l_nx += p_c2.r;
    l_dist = Math.sqrt(l_nx * l_nx + l_ny * l_ny);
  }
  // Kugeln kollidieren, wenn der Abstand kleiner gleich der Summe
  // der Radien beider Kugeln ist.
  if (l_dist <= p_c1.r + p_c2.r)
  { // Normalenvektor wird normalisiert: |l_n| = 1
    l_nx /= l_dist;
    l_ny /= l_dist;
    // Tangentialvektor (senkrecht zu Normalenvektor, zwischen
    // beiden Kugeln)
    let l_tx = l_ny;
    let l_ty = -l_nx;
    // Summe der Massen beider Kugeln
    let l_sm1m2 = p_c1.m + p_c2.m;
    // Überlappung der beiden Kugeln
    let l_overlap = (p_c1.r + p_c2.r) - l_dist;
```

```

// Verschieben der beiden Kugeln entlang der Normalen,
// so dass sie sich nicht mehr überlappen!
// Die Massen werden berücksichtigt. Schwerere Kreise werden
// weniger weit verschoben.
l_overlap += 2;      /* fight penetration */
l_overlap *= 1.001; /* fight penetration */
p_c1.x = p_c1.x - l_nx * l_overlap * (p_c2.m / l_sm1m2);
p_c1.y = p_c1.y - l_ny * l_overlap * (p_c2.m / l_sm1m2);
p_c2.x = p_c2.x + l_nx * l_overlap * (p_c1.m / l_sm1m2);
p_c2.y = p_c2.y + l_ny * l_overlap * (p_c1.m / l_sm1m2);
// Zerlegung der Geschwindigkeitsvektoren in Normalen- und
// Tangentialanteil: v=sn*n+st*t, wobei
// v Vektor, n Normalenvektor, t Tangentialvektor und
// sn, st zwei skalare Werte sind.
// Es gilt: v*n = sn*(n*n)+st*(t*n) = sn, da t*n=0 und n*n = 1
// Es gilt: v*t = sn*(n*t)+st*(t*t) = st, da t*n=0 und t*t = 1
// Also ist: sn = v*n und st=v*t
// Ball 1: Zerlegung des Geschwindigkeitsvektors in n- und t-
Anteil
let l_sn1 = l_nx * p_c1.vx + l_ny * p_c1.vy;
let l_st1 = l_tx * p_c1.vx + l_ty * p_c1.vy;
let l_n1x = l_nx * l_sn1; // Normalenvektor-Anteil von p_c1.vx
let l_n1y = l_ny * l_sn1; // Normalenvektor-Anteil von p_c1.vy
let l_t1x = l_tx * l_st1; // Tangentialvektor-Anteil von p_c1.vx
let l_t1y = l_ty * l_st1; // Tangentialvektor-Anteil von p_c1.vy
// Ball 2: Zerlegung des Geschwindigkeitsvektors in n- und t-
Anteil
let l_sn2 = l_nx * p_c2.vx + l_ny * p_c2.vy;
let l_st2 = l_tx * p_c2.vx + l_ty * p_c2.vy;
let l_n2x = l_nx * l_sn2; // Normalenvektor-Anteil von p_c2.vx
let l_n2y = l_ny * l_sn2; // Normalenvektor-Anteil von p_c2.vy
let l_t2x = l_tx * l_st2; // Tangentialvektor-Anteil von p_c2.vx
let l_t2y = l_ty * l_st2; // Tangentialvektor-Anteil von p_c2.vy
// Der Impulserhaltungssatz
//  $m_1*v_1 + m_2*v_2 = m_1*v_1' + m_2*v_2'$ 
// (wobei m1, m2 = Massen der Körper
// und v1, v2, v1', v2' die Geschwindigkeiten)
// und der Energieerhaltungssatz
//  $0,5*m_1*v_1^2 + 0,5*m_2*v_2^2 = 0,5*m_1*v_1'^2 + 0,5*m_2*v_2'^2$ 
// führen nach einfachen mathematischen Umformungen zu
// folgenden Beziehungen (für den eindimensionalen Fall):
//  $v_1' = 2*(m_1*v_1+m_2*v_2)/(m_1+m_2) - v_1$ 
//  $v_2' = 2*(m_1*v_1+m_2*v_2)/(m_1+m_2) - v_2$ 
//  $2*(m_1*v_1+m_2*v_2)/(m_1+m_2)$  ist die Geschwindigkeit des
// gemeinsamen Schwerpunktes
// (center of gravity).

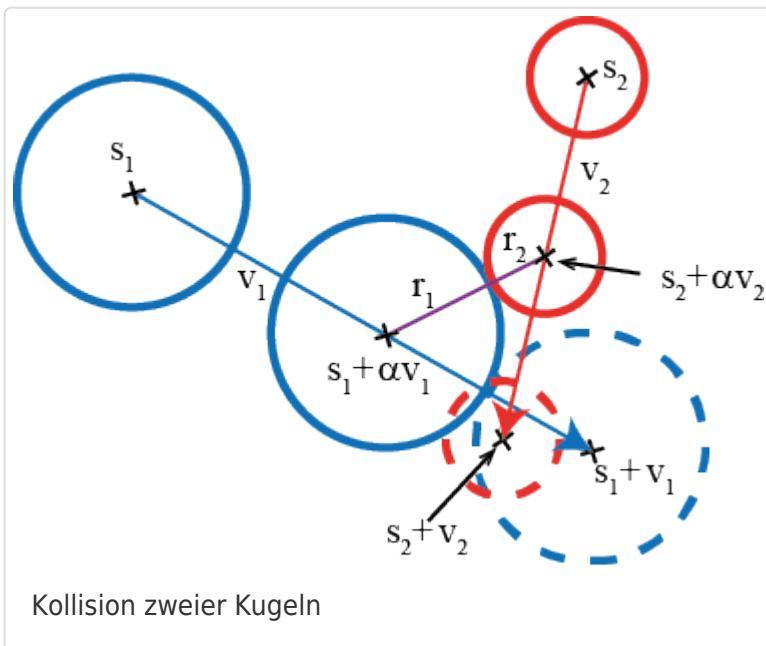
```

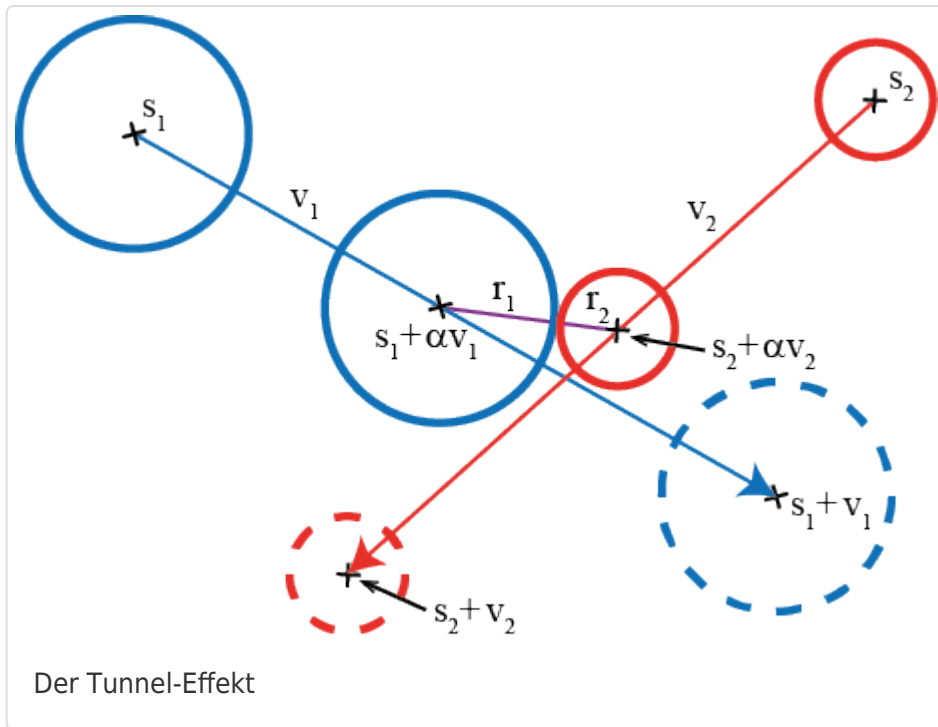
```

// Im zweidimensionalen Fall gilt, dass die Kollision entlang
// der Normalen erfolgt. Die tangentialen Anteile der der
// Bewegungsrichtungen werden unverändert übernommen.
let l_vcgx = 2*(p_c1.m * l_n1x + p_c2.m * l_n2x) / l_sm1m2;
let l_vcgy = 2*(p_c1.m * l_n1y + p_c2.m * l_n2y) / l_sm1m2;
p_c1.vx = l_vcgx - l_n1x + l_t1x;
p_c1.vy = l_vcgy - l_n1y + l_t1y;
p_c2.vy = l_vcgx - l_n2x + l_t2x;
p_c2.vy = l_vcgy - l_n2y + l_t2y;
//// Alternative Berechnung:
// Differenz der Massen beide Kugeln
// let l_dm2m1 = p_c2.m - p_c1.m;
// p_c1.vx = (l_n2x*p_c2.m*2-l_n1x*l_dm2m1)/l_sm1m2 + l_t1x;
// p_c1.vy = (l_n2y*p_c2.m*2-l_n1y*l_dm2m1)/l_sm1m2 + l_t1y;
// p_c2.vx = (l_n1x*p_c1.m*2+l_n2x*l_dm2m1)/l_sm1m2 + l_t2x;
// p_c2.vy =(l_n1y*p_c1.m*2+l_n2y*l_dm2m1)/l_sm1m2 + l_t2y;
return true;
}
else
{ return false; }
}

```

3 A-priori-Kollisionserkennung von zwei Kugeln





4 Quellen

1. **Kowarschick (MMProg):** Wolfgang Kowarschick; Vorlesung „Multimedia-Programmierung“; Hochschule: Hochschule Augsburg; Adresse: Augsburg; Web-Link; 2018; Quellengüte: 3 (Vorlesung)

Kategorie:
Spielephysik

Diese Seite wurde zuletzt am 23. November 2018 um 19:52 Uhr bearbeitet.

Inhalt verfügbar unter [CC BY-NC-SA 4.0](#), falls Dokument nach dem 5. 3. 2011 erstellt wurde, sonst [CC BY-SA DE 3.0](#).

