

Bezugssysteme und Einführung

Einführung in Physik Engines

Physics Engines

Physics Engines sind Softwarekomponenten, die physikalische Effekte in Computerprogrammen (z.B. Spielen) simulieren. In diesem Modul geht es um:

- Physikalische Modellierung in Unity
- Verstehen physikalischer Grundprinzipien für die korrekte Anwendung in Unity
- Kopplung von Physiksimitatoren (wie PhysX) mit realistischen Parametern

Selbst wenn Unity's PhysX viele Funktionen automatisch berechnet, benötigt man ein fundiertes Verständnis der Physik, um realistische Simulationen zu erstellen.

Die Modellbildung in Unity folgt diesem Ablauf:

1. Wirklichkeit → Physikalisches Modell
2. Physikalisches Modell → Mathematisches Modell
3. Mathematisches Modell → Numerisches Modell
4. Numerisches Modell → Modellerte Werte
5. Diese können dann mit experimentellen Daten verglichen werden

Bezugssysteme in der Mechanik

Bezugssystem

Ein Bezugssystem definiert:

- Einen Nullpunkt im Raum
- Die Richtungen der Koordinatenachsen (x, y, z)
- Eine Zeitmessung

Dadurch wird die Position eines Körpers eindeutig durch einen Ortsvektor \vec{r} beschrieben.

Vektoren

Ein Vektor ist eine physikalische Größe mit Betrag und Richtung.

- Darstellung: \vec{r} (mit Pfeil über dem Symbol)
- Betrag: $|\vec{r}| = r$ (ohne Pfeil)

- In Koordinatendarstellung: $\vec{r} = \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix}$
- Einheitsvektor (Betrag = 1): $\vec{e}_r = \frac{\vec{r}}{|\vec{r}|}$

Rechenregeln für Vektoren

- Addition: $\vec{r}_1 + \vec{r}_2 = \begin{pmatrix} r_{x1} \\ r_{y1} \\ r_{z1} \end{pmatrix} + \begin{pmatrix} r_{x2} \\ r_{y2} \\ r_{z2} \end{pmatrix} = \begin{pmatrix} r_{x1} + r_{x2} \\ r_{y1} + r_{y2} \\ r_{z1} + r_{z2} \end{pmatrix}$
- Skalarprodukt (ergibt einen Skalar): $s = \vec{r}_1 \cdot \vec{r}_2 = |\vec{r}_1| \cdot |\vec{r}_2| \cdot \cos \angle(\vec{r}_1, \vec{r}_2) = r_{x1}r_{x2} + r_{y1}r_{y2} + r_{z1}r_{z2}$
- Kreuzprodukt (ergibt einen Vektor): $\vec{r}_1 \times \vec{r}_2 = \begin{pmatrix} r_{y1}r_{z2} - r_{z1}r_{y2} \\ r_{z1}r_{x2} - r_{x1}r_{z2} \\ r_{x1}r_{y2} - r_{y1}r_{x2} \end{pmatrix}$

Unity Bezugssystem

Ein Unterschied zwischen Unity und den in der Physik üblichen Bezugssystemen:

- Physik und Mathematik verwenden üblicherweise ein Rechtssystem
- Unity verwendet ein Linkssystem
- In Unity zeigt die positive y-Achse nach oben (nicht die z-Achse)

Dies hat Auswirkungen auf die Berechnung von Kreuzprodukten und die Interpretation von Rotationen.

SI-Einheiten

Wichtige SI-Einheiten in der Mechanik:

- Länge in Meter (m)
- Masse in Kilogramm (kg)
- Zeit in Sekunden (s)
- Kraft in Newton (N = kg · m/s²)

Es ist wichtig, in Unity konsequent SI-Einheiten zu verwenden und bei allen Werten entsprechende Einheiten anzugeben.

Konventionen für Unity-Code

Deklaration von physikalischen Größen

```
1 // Korrekte Deklaration physikalischer Groessen in Unity
2 public float initialVelocity = 6.0f; // in m/s
3 public float mass = 1.0f; // in kg
4 public float springConstant = 10.0f; // in N/m
```

Abfrage von Vektoren

```
1 // Abfrage von Position und Geschwindigkeit
2 Vector3 position = rigidBody.position; // in m
3 Vector3 velocity = rigidBody.velocity; // in m/s
```

Berechnung von Kräften

```
1 // Beispiel: Federkraft berechnen
2 Vector3 springForce = -springConstant * (position - equilibriumPosition);
3 rigidBody.AddForce(springForce); // Kraft in N hinzufuegen
```

Für die Umrechnung der Geschwindigkeit von km/h in m/s teilt man durch 3,6: $v[m/s] = \frac{v[km/h]}{3,6}$

Beispiel: 72 km/h = 72 / 3,6 = 20 m/s

Kinematik der Translation

Grundlagen der Kinematik

Kinematik

Die Kinematik beschreibt die Bewegung eines Körpers, ohne auf deren Ursachen einzugehen. Die Bewegung eines Körpers ist vollständig beschrieben durch:

- seinen Ort (Vektor \vec{r})
- seine Geschwindigkeit (\vec{v})
- seine Beschleunigung (\vec{a})

Diese drei Größen hängen durch Ableiten bzw. Integrieren zusammen.

Zusammenhänge zwischen Ort Geschwindigkeit und Beschleunigung

- Geschwindigkeit = Ableitung des Ortes nach der Zeit:
 $\vec{v} = \frac{d\vec{r}}{dt}$
- Beschleunigung = Ableitung der Geschwindigkeit nach der Zeit:
 $\vec{a} = \frac{d\vec{v}}{dt}$
- Ort = Integral der Geschwindigkeit nach der Zeit:
 $\vec{r} = \int \vec{v} dt$
- Geschwindigkeit = Integral der Beschleunigung nach der Zeit:
 $\vec{v} = \int \vec{a} dt$

Mittlere Geschwindigkeit und Beschleunigung

Mittlere Geschwindigkeit

Die mittlere Geschwindigkeit ist die Änderung des Ortes dividiert durch die dafür benötigte Zeit:

$$\bar{v}_x = \frac{\Delta r_x}{\Delta t} \tag{1}$$

Sie stellt den Durchschnittswert über das betrachtete Zeitintervall dar.

Mittlere Beschleunigung

Die mittlere Beschleunigung ist die Änderung der Geschwindigkeit dividiert durch die dafür benötigte Zeit:

$$\bar{a}_x = \frac{\Delta v_x}{\Delta t} \tag{2}$$

Differenzenquotient vs. Differentialquotient

- Der Differenzenquotient (mittlere Geschwindigkeit) ist eine Approximation über ein endliches Zeitintervall:
 $\frac{\Delta r_x}{\Delta t}$
- Der Differentialquotient (Momentangeschwindigkeit) ist der Grenzwert für ein infinitesimal kleines Zeitintervall: $\lim_{\Delta t \rightarrow 0} \frac{\Delta r_x}{\Delta t} = \frac{dr_x}{dt}$
- In Unity wird mit fixen Zeitschritten $\Delta t = 20$ ms gerechnet, was einer Abtastfrequenz von $f_{sample} = 50$ Hz entspricht

Momentangeschwindigkeit und -beschleunigung

Momentangeschwindigkeit

Die Momentangeschwindigkeit zur Zeit t_0 ist definiert als:

$$\vec{v}(t_0) = \lim_{t_1 \rightarrow t_0} \frac{\Delta \vec{r}}{t_1 - t_0} = \frac{d\vec{r}}{dt} \tag{3}$$

Sie entspricht geometrisch der Steigung der Tangente im Punkt $(t_0, r_x(t_0))$.

Der Betrag der Geschwindigkeit wird oft als Schnelligkeit bezeichnet:

$$|\vec{v}| = \sqrt{v_x^2 + v_y^2 + v_z^2} \tag{4}$$

Bei gleichbleibender Schnelligkeit kann sich dennoch die Richtung der Geschwindigkeit ändern, z.B. bei einer Kreisbewegung.

Fläche unter dem Geschwindigkeits-Zeit-Diagramm

Bei einer Bewegung mit variablem $v(t)$ berechnet sich die zurückgelegte Strecke als Fläche unter der v - t -Kurve:

$$\Delta x = \int_{t_1}^{t_2} v(t) dt \tag{5}$$

Integration und Differentiation

Ableitungsregeln

- Konstante Summanden: $\frac{d}{dt}(C) = 0$
- Potenzfunktionen: $\frac{d}{dt}(at^n) = a \cdot n \cdot t^{n-1}$
- Exponentialfunktion: $\frac{d}{dx}(e^x) = e^x$
- Logarithmus: $\frac{d}{dx}(\ln x) = \frac{1}{x}$
- Sinus/Kosinus: $\frac{d}{dx}(\sin x) = \cos x, \frac{d}{dx}(\cos x) = -\sin x$

Regeln für zusammengesetzte Funktionen

- Summenregel: $\frac{d}{dt}(f(t) + g(t)) = \frac{df}{dt} + \frac{dg}{dt}$
- Produktregel: $\frac{d}{dt}(f(t) \cdot g(t)) = \frac{df}{dt} \cdot g(t) + f(t) \cdot \frac{dg}{dt}$
- Kettenregel: $\frac{d}{dt}(f(g(t))) = \frac{df}{dg} \cdot \frac{dg}{dt}$

Berechnung von Bewegungen mit konstanter Beschleunigung

Gegebene Größen

- Anfangsposition r_0
- Anfangsgeschwindigkeit v_0
- Konstante Beschleunigung a

Schritte zur Berechnung

1. Geschwindigkeit in Abhängigkeit von der Zeit bestimmen:

$$v(t) = v_0 + at \tag{6}$$

2. Position in Abhängigkeit von der Zeit bestimmen:

$$r(t) = r_0 + v_0 t + \frac{1}{2}at^2 \tag{7}$$

3. Alternative Formel bei bekannter Strecke (ohne Zeit):

$$v^2 = v_0^2 + 2a(r - r_0) \tag{8}$$

Bewegung in Unity implementieren

```
1 // Implementierung von Bewegungen mit konstanter Beschleunigung
2 void FixedUpdate() {
3     // Aktuelle Zeit seit Start
4     currentTime += Time.deltaTime;
5
6     // Aktuelle Geschwindigkeit nach  $v = v_0 + a \cdot t$  berechnen
7     float currentVelocity = initialVelocity + acceleration * currentTime;
8
9     // Bewegung mit aktueller Geschwindigkeit
10    Vector3 displacement = new Vector3(currentVelocity, 0, 0) * Time.deltaTime;
11    transform.position += displacement;
12
13    // Alternative: Direkte Berechnung der Position mit  $r = r_0 + v_0 \cdot t + 0.5 \cdot a \cdot t^2$ 
14    // Vector3 newPosition = initialPosition + initialVelocity * currentTime +
15    //                        0.5f * acceleration * currentTime * currentTime;
16    // transform.position = newPosition;
17 }
```

Freier Fall

Ein Körper fällt aus der Höhe r_0 mit Anfangsgeschwindigkeit $v_0 = 0$.

- Beschleunigung: $a(t) = -g$ ($g = 9.81 \text{ m/s}^2$)
- Geschwindigkeit: $v(t) = -gt$
- Position: $r(t) = r_0 - \frac{1}{2}gt^2$

Alternativ: Ein Körper wird mit Anfangsgeschwindigkeit v_0 nach oben geworfen:

- Maximale Höhe: $h_{max} = \frac{v_0^2}{2g}$
- Zeit bis zum höchsten Punkt: $t_{max} = \frac{v_0}{g}$
- Gesamtflugzeit: $t_{gesamt} = \frac{2v_0}{g}$

Grundlagen der Dynamik

Dynamik

Die Dynamik beschäftigt sich mit den Ursachen von Bewegungen, also den Kräften, die auf einen Körper wirken. Sie baut auf der Kinematik auf und erweitert diese um die Betrachtung der wirkenden Kräfte.

Newton'sche Axiome

Isaac Newton formulierte die drei grundlegenden Gesetze der Bewegung:

1. Trägheitsgesetz: Ein Körper bleibt im Zustand der Ruhe oder der gleichförmigen geradlinigen Bewegung, solange keine Kraft auf ihn wirkt.
2. Bewegungsgesetz: Die Änderung der Bewegung ist proportional zur einwirkenden Kraft und erfolgt in Richtung der Kraft.
3. Wechselwirkungsgesetz: Übt ein Körper auf einen anderen eine Kraft aus (actio), so wirkt eine gleich große, entgegengesetzte Kraft zurück (reactio).

Ein viertes Prinzip ist das Superpositionsprinzip: Kräfte addieren sich vektoriell.

Erstes Newton'sches Gesetz (Trägheitsgesetz)

Trägheitsgesetz

Das erste Newton'sche Gesetz lässt sich mathematisch ausdrücken als:

$$\vec{F} = 0 \Rightarrow \vec{v} = \text{const.} \quad (9)$$

Dies bedeutet auch:

$$\vec{F} = 0 \Rightarrow \vec{a} = \frac{d\vec{v}}{dt} = \frac{d}{dt}(\text{const.}) = 0 \quad (10)$$

Das Gesetz gilt nur in sogenannten Inertialsystemen: Bezugssystemen ohne Beschleunigung oder Rotation.

Trägheitsgesetz auf der Luftkissenbahn

Ein Gleiter auf einer Luftkissenbahn bewegt sich ohne Reibung mit konstanter Geschwindigkeit weiter, wenn keine Kraft auf ihn wirkt. Dies ist eine direkte Demonstration des Trägheitsgesetzes.

Das gleiche Prinzip gilt für einen Stein, der von der Spitze eines fahrenden Schiffes fällt - er behält seine horizontale Geschwindigkeit bei und landet am Fuß des Mastes, nicht dahinter oder davor.

Zweites Newton'sches Gesetz (Bewegungsgesetz)

Impuls

Der Impuls \vec{p} eines Körpers ist das Produkt aus seiner Masse und seiner Geschwindigkeit:

$$\vec{p} = m \cdot \vec{v} \quad (11)$$

Der Impuls ist eine vektorielle Größe mit der Einheit $\text{kg} \cdot \text{m/s}$.

Bewegungsgesetz

Das zweite Newton'sche Gesetz in seiner allgemeinen Form:

$$\vec{F} = \frac{d\vec{p}}{dt} \quad (12)$$

Für Körper mit konstanter Masse vereinfacht sich dies zu:

$$\vec{F} = m \cdot \vec{a} \quad (13)$$

In integraler Form:

$$\vec{p} = \vec{p}_0 + \int_0^t \vec{F}(t) dt \quad (14)$$

wobei $\int \vec{F} dt$ als Kraftstoß bezeichnet wird.

Berechnung von Kräften und Beschleunigungen

Berechnung der Beschleunigung aus Kräften

- Alle auf den Körper wirkenden Kräfte identifizieren
- Kräfte vektoriell addieren zur resultierenden Kraft \vec{F}_{res}
- Beschleunigung berechnen: $\vec{a} = \frac{\vec{F}_{res}}{m}$

Beispiel: Fahrstuhl

- Fahrstuhl in Ruhe: $F_{Seil} - F_G = 0 \Rightarrow F_{Seil} = m \cdot g$
- Fahrstuhl beschleunigt nach oben: $F_{Seil} - F_G = m \cdot a \Rightarrow F_{Seil} = m \cdot (g + a)$
- Fahrstuhl beschleunigt nach unten: $F_{Seil} - F_G = -m \cdot a \Rightarrow F_{Seil} = m \cdot (g - a)$

Drittes Newton'sches Gesetz (Wechselwirkungsgesetz)

Wechselwirkungsgesetz

Das dritte Newton'sche Gesetz lautet mathematisch:

$$\vec{F}_{12} = -\vec{F}_{21} \quad (15)$$

wobei \vec{F}_{12} die Kraft bezeichnet, die Körper 1 auf Körper 2 ausübt, und \vec{F}_{21} die Kraft, die Körper 2 auf Körper 1 ausübt.

Merkmale von Kräftepaaren

Kräftepaare des Wechselwirkungsgesetzes haben folgende Eigenschaften:

1. Gleicher Betrag, entgegengesetzte Richtung
2. Greifen an verschiedenen Körpern an
3. Haben die gleiche physikalische Ursache

Anwendungen des Wechselwirkungsgesetzes

- Gravitationskraft: Die Erde zieht einen Menschen an, aber der Mensch zieht auch die Erde an (mit der gleichen Kraft)
- Bremsen eines Autos: Das Auto stößt die Straße nach vorn, die Straße stößt das Auto nach hinten
- Zwei verbundene Boote: Wenn eine Person in einem Boot das andere Boot an einer Leine zu sich zieht, bewegen sich beide Boote aufeinander zu

Superpositionsprinzip

Superpositionsprinzip

Das Superpositionsprinzip besagt, dass sich mehrere Kräfte $\vec{F}_1, \vec{F}_2, \dots, \vec{F}_n$, die auf einen Punkt wirken, vektoriell zu einer resultierenden Kraft \vec{F} addieren:

$$\vec{F} = \sum_{i=1}^n \vec{F}_i \quad (16)$$

Für jede Komponente gilt entsprechend:

$$F_x = \sum_{i=1}^n F_{xi} \quad F_y = \sum_{i=1}^n F_{yi} \quad F_z = \sum_{i=1}^n F_{zi} \quad (17)$$

Kräfte "freischneiden"

Schritte zum Freischneiden

1. Zeichne den zu untersuchenden Körper isoliert
2. Zeichne alle auf den Körper wirkenden Kräfte als Vektoren ein
3. Bestimme die resultierende Kraft: $\vec{F}_{res} = \sum \vec{F}_i$
4. Berechne die Beschleunigung: $\vec{a} = \frac{\vec{F}_{res}}{m}$

Beispiel: Kiste auf schiefer Ebene

- Gewichtskraft: $\vec{F}_G = m \cdot \vec{g}$ (nach unten)
- Normalkraft: \vec{F}_N (senkrecht zur Ebene)
- Reibungskraft: $\vec{F}_R = \mu \cdot \vec{F}_N$ (parallel zur Ebene, entgegen der Bewegungsrichtung)
- Hangabtriebskraft: $F_{Hang} = m \cdot g \cdot \sin \alpha$ (parallel zur Ebene nach unten)

Kräfte in Unity implementieren

```
1 // Anwenden einer Kraft in Unity
2 void FixedUpdate() {
3     // Gravitationskraft berechnen
4     Vector3 gravityForce = new Vector3(0, -9.81f * mass, 0);
5
6     // Normalkraft berechnet Unity automatisch bei Kollisionen
7
8     // Hangabtriebskraft bei schiefer Ebene mit Winkel alpha
9     float alpha = 30f * Mathf.Deg2Rad; // 30 Grad in Radian
10    float hangForce = mass * 9.81f * Mathf.Sin(alpha);
11    Vector3 hangForceVector = new Vector3(hangForce, 0, 0);
12
13    // Reibungskraft
14    float frictionCoeff = 0.3f;
15    Vector3 frictionForce = -frictionCoeff * mass * 9.81f * Mathf.Cos(alpha) *
        rigidBody.velocity.normalized;
16
17    // Alle Kräfte anwenden
18    rigidBody.AddForce(gravityForce + hangForceVector + frictionForce);
19 }
```

Kräfte

Arten von Kräften

Grundlegende Wechselwirkungen

In der Physik gibt es vier fundamentale Wechselwirkungen:

1. Gravitation (Anziehung zwischen Massen)
2. Elektromagnetische Kraft (Kräfte zwischen elektrischen Ladungen)
3. Starke Kernkraft (hält den Atomkern zusammen)
4. Schwache Kernkraft (verantwortlich für radioaktiven Beta-Zerfall)

Für die makroskopische Mechanik sind hauptsächlich die Gravitation und die elektromagnetischen Kräfte relevant. Die starke und schwache Kernkraft wirken nur auf atomarer Ebene.

Kraft

Eine Kraft ist ein Einfluss, der den Bewegungszustand eines Körpers ändert. Im SI-System wird sie in Newton (N) gemessen:

$$1\text{ N} = 1\frac{\text{kg} \cdot \text{m}}{\text{s}^2} \tag{18}$$

Kräfte ändern den Impuls eines Körpers gemäß der Formel:

$$\vec{F} = \frac{d\vec{p}}{dt} \tag{19}$$

Gravitationskraft

Gravitationsgesetz

Das Newton'sche Gravitationsgesetz beschreibt die Anziehungskraft zwischen zwei Massen m_1 und m_2 , die im Abstand R voneinander entfernt sind:

$$F_G = G \cdot \frac{m_1 \cdot m_2}{R^2} \tag{20}$$

Dabei ist G die Gravitationskonstante mit dem Wert:

$$G = 6.67 \cdot 10^{-11} \frac{\text{m}^3}{\text{kg} \cdot \text{s}^2} \tag{21}$$

Die Kraft wirkt entlang der Verbindungslinie der beiden Massen und ist eine Anziehungskraft.

Erdbeschleunigung

Die Gewichtskraft F_G eines Körpers der Masse m auf der Erdoberfläche ist:

$$F_G = m \cdot g \tag{22}$$

Die Erdbeschleunigung g lässt sich aus dem Gravitationsgesetz herleiten:

$$g = G \cdot \frac{M_{\text{Erde}}}{R_{\text{Erde}}^2} \approx 9.81 \frac{\text{m}}{\text{s}^2} \tag{23}$$

Mit zunehmender Höhe h nimmt die Fallbeschleunigung ab:

$$g(h) = G \cdot \frac{M_{\text{Erde}}}{(R_{\text{Erde}} + h)^2} \tag{24}$$

Federkraft

Federkraft (Hooke'sches Gesetz)

Für eine lineare Feder gilt das Hooke'sche Gesetz:

$$\vec{F} = -k \cdot \vec{x} \tag{25}$$

Dabei ist:

- k : die Federkonstante in N/m
- \vec{x} : die Auslenkung der Feder aus ihrer Ruhelage
- Das negative Vorzeichen bedeutet, dass die Kraft der Auslenkung entgegengerichtet ist

Spannenergie einer Feder

Die in einer gespannten Feder gespeicherte potentielle Energie beträgt:

$$E_{\text{spann}} = \frac{1}{2} \cdot k \cdot x^2 \tag{26}$$

Diese Energie kann in kinetische Energie umgewandelt werden, wenn die Feder entspannt wird.

Harmonischer Oszillator

Ein harmonischer Oszillator ist ein System, bei dem die rücktreibende Kraft proportional zur Auslenkung ist. Beispiele sind eine Masse an einer Feder oder ein Pendel bei kleinen Auslenkungen.

Die Bewegungsgleichung eines harmonischen Oszillators lautet:

$$\frac{d^2x}{dt^2} = -\frac{k}{m} \cdot x \tag{27}$$

Die Lösung ist eine harmonische Schwingung mit der Kreisfrequenz ω :

$$x(t) = x_0 \cdot \cos(\omega t) \tag{28}$$

wobei $\omega = \sqrt{\frac{k}{m}}$ und die Schwingungsdauer $T = \frac{2\pi}{\omega}$ beträgt.

Simulation eines harmonischen Oszillators in Unity

```
1 // Harmonischer Oszillator
2 void FixedUpdate() {
3     // Federkraft berechnen (F = -k*x)
4     float springForce = -springConstant * rigidbody.position.x;
5
6     // Kraft anwenden
7     rigidbody.AddForce(new Vector3(springForce, 0f, 0f));
8
9     // Zeit aktualisieren
10    currentTimeStep += Time.deltaTime;
11 }
```

Reibungskräfte

Arten der Reibung

Man unterscheidet zwischen:

- Äußerer Reibung (Kontaktflächen von sich berührenden Festkörpern):
 - Haftreibung
 - Gleitreibung
 - Rollreibung, Wälzreibung, Bohrreibung, Seilreibung
- Innerer Reibung (zwischen benachbarten Teilchen bei Verformungen innerhalb von Festkörpern, Flüssigkeiten und Gasen)

Trockene Reibung

Für trockene Reibung (Coulomb-Reibung) gilt:

$$\vec{F}_R = \mu \cdot \vec{F}_N \quad (29)$$

Dabei ist:

- μ : Reibungskoeffizient (dimensionslos)
 - \vec{F}_N : Normalkraft
 - Die Richtung der Reibungskraft ist stets entgegengesetzt zur Bewegungsrichtung bzw. zur Zugkraft
- Man unterscheidet zwischen:
- Haftreibung: $\vec{F}_{\text{Haft}} \leq \mu_{\text{Haft}} \cdot \vec{F}_N$ (gleichet äußere Kraft bis zu einem Maximalwert aus)
 - Gleitreibung: $\vec{F}_{\text{Gleit}} = \mu_{\text{Gleit}} \cdot \vec{F}_N$ (konstant bei gegebener Normalkraft)
 - Dabei gilt in der Regel: $\mu_{\text{Haft}} > \mu_{\text{Gleit}}$

Viskose Reibung

Für die Reibung in Flüssigkeiten und Gasen gelten je nach Strömungsregime:

- Laminare Strömung (Stokes'sche Reibung für eine Kugel):

$$\vec{F}_R = -6 \cdot \pi \cdot \eta \cdot r \cdot v \cdot \vec{e}_v \quad (30)$$

wobei η die Viskosität des Mediums ist

- Turbulente Strömung:

$$\vec{F}_R = -\frac{1}{2} \cdot \rho \cdot A \cdot c_w \cdot v^2 \cdot \vec{e}_v \quad (31)$$

wobei ρ die Dichte des Mediums, A die Stirnfläche und c_w der Widerstandsbeiwert ist

Reibung in Unity implementieren

Trockene Reibung

```
1 // Trockene Reibung implementieren
2 void FixedUpdate() {
3     // Normale Kraft berechnen (Gewichtskraft bei horizontaler Flaeche)
4     float normalForce = rigidbody.mass * 9.81f;
5
6     // Reibungskraft berechnen
7     float frictionForce = frictionCoefficient * normalForce;
8
9     // Richtung der Geschwindigkeit bestimmen
10    Vector3 velocityDirection = rigidbody.velocity.normalized;
11
12    // Reibungskraft nur anwenden, wenn Objekt sich bewegt
13    if (rigidbody.velocity.magnitude > 0.01f) {
14        // Reibungskraft entgegen der Bewegungsrichtung
15        Vector3 frictionVector = -velocityDirection * frictionForce;
16        rigidbody.AddForce(frictionVector);
17    }
18 }
```

Luftwiderstand (turbulente Reibung)

```
1 // Luftwiderstand implementieren
2 void FixedUpdate() {
3     // Parameter fuer Luftwiderstand
4     float airDensity = 1.2f; // kg/m^3
5     float dragCoefficient = 0.5f; // dimensionslos
6     float frontalArea = 1.0f; // m^2
7
8     // Aktuelle Geschwindigkeit und Betrag
9     Vector3 velocity = rigidbody.velocity;
10    float velocityMagnitude = velocity.magnitude;
11
12    // Luftwiderstandskraft berechnen
13    float dragForceMagnitude = 0.5f * airDensity * frontalArea *
14        dragCoefficient * velocityMagnitude * velocityMagnitude;
15
16    // Richtung entgegen der Bewegung
17    Vector3 dragForce = -velocity.normalized * dragForceMagnitude;
18
19    // Kraft anwenden
20    rigidbody.AddForce(dragForce);
21 }
```

Trägheitskräfte

Beschleunigte Bezugssysteme

In beschleunigten Bezugssystemen treten Trägheitskräfte (auch Scheinkräfte genannt) auf, um die Newton'schen Gesetze weiterhin anwenden zu können.

Beispiele:

- Translatorische Trägheitskraft bei linearer Beschleunigung:

$$\vec{F}_{\text{Trägheit}} = -m \cdot \vec{a}_{\text{System}} \quad (32)$$

- Zentrifugalkraft bei Rotation:

$$\vec{F}_{\text{Zentrifugal}} = m \cdot \omega^2 \cdot r \cdot \vec{e}_r \quad (33)$$

- Corioliskraft bei Bewegung in einem rotierenden System:

$$\vec{F}_{\text{Coriolis}} = 2 \cdot m \cdot \vec{v} \times \vec{\omega} \quad (34)$$

Trägheitskräfte im Alltag

- Im bremsenden Zug fühlt man sich nach vorne gedrückt
- In einer Kurve wird man nach außen gedrückt (Zentrifugalkraft)
- Die Corioliskraft beeinflusst Winde und Meeresströmungen auf der Erde (auf der Nordhalbkugel werden sie nach rechts, auf der Südhalbkugel nach links abgelenkt)

Trägheitskräfte sind keine echten "Kräfte im Sinne von Wechselwirkungen zwischen Körpern, sondern entstehen durch die Wahl des Bezugssystems. In einem Inertialsystem existieren sie nicht.

Impuls und Stoßgesetze

Impuls

Impuls

Der Impuls \vec{p} eines Körpers ist das Produkt aus seiner Masse und seiner Geschwindigkeit:

$$\vec{p} = m \cdot \vec{v} \quad (35)$$

Der Impuls ist eine vektorielle Größe mit der Einheit kg · m/s.

Impulserhaltung

Das Prinzip der Impulserhaltung besagt, dass in einem abgeschlossenen System der Gesamtimpuls konstant bleibt, wenn keine äußeren Kräfte wirken:

$$\sum \vec{p}_{\text{vorher}} = \sum \vec{p}_{\text{nachher}} \quad (36)$$

In Komponenten ausgedrückt für zwei Körper:

$$m_1 \cdot v_{1,\text{vorher}} + m_2 \cdot v_{2,\text{vorher}} = m_1 \cdot v_{1,\text{nachher}} + m_2 \cdot v_{2,\text{nachher}} \quad (37)$$

Die Impulserhaltung gilt auch bei dissipativen Vorgängen wie inelastischen Stößen, bei denen Energie verloren geht.

Kraftstoß

Der Kraftstoß \vec{I} ist das Zeitintegral der Kraft:

$$\vec{I} = \int_{t_1}^{t_2} \vec{F}(t) dt \quad (38)$$

Der Kraftstoß ist gleich der Impulsänderung:

$$\vec{I} = \Delta \vec{p} = \vec{p}_{\text{nachher}} - \vec{p}_{\text{vorher}} \quad (39)$$

Für eine konstante Kraft während der Zeitspanne Δt gilt:

$$\vec{I} = \vec{F} \cdot \Delta t \quad (40)$$

Stöße

Elastischer Stoß

Bei einem elastischen Stoß bleiben sowohl der Gesamtimpuls als auch die Gesamtenergie erhalten. Es gilt:

$$m_1 \cdot v_{1,\text{vorher}} + m_2 \cdot v_{2,\text{vorher}} = m_1 \cdot v_{1,\text{nachher}} + m_2 \cdot v_{2,\text{nachher}} \quad (41)$$

$$\frac{1}{2} m_1 \cdot v_{1,\text{vorher}}^2 + \frac{1}{2} m_2 \cdot v_{2,\text{vorher}}^2 = \frac{1}{2} m_1 \cdot v_{1,\text{nachher}}^2 + \frac{1}{2} m_2 \cdot v_{2,\text{nachher}}^2 \quad (42)$$

Inelastischer Stoß

Bei einem inelastischen Stoß bleibt nur der Gesamtimpuls erhalten, während die mechanische Energie teilweise in andere Energieformen (z.B. Wärme) umgewandelt wird.

Im Extremfall des vollständig inelastischen Stoßes "kleben" die Körper nach dem Stoß zusammen und bewegen sich mit einer gemeinsamen Geschwindigkeit v_{nachher} :

$$v_{\text{nachher}} = \frac{m_1 \cdot v_{1,\text{vorher}} + m_2 \cdot v_{2,\text{vorher}}}{m_1 + m_2} \quad (43)$$

Berechnung von Stößen in 1D

Elastischer Stoß

Gegeben: Massen m_1 und m_2 , Anfangsgeschwindigkeiten v_1 und v_2

1. Schwerpunktgeschwindigkeit berechnen:

$$v_{Spt} = \frac{m_1 \cdot v_1 + m_2 \cdot v_2}{m_1 + m_2} \quad (44)$$

2. Geschwindigkeiten nach dem Stoß berechnen (Spiegelung an der Schwerpunktgeschwindigkeit):

$$u_1 = v_{Spt} - (v_1 - v_{Spt}) = 2v_{Spt} - v_1 \quad (45)$$

$$u_2 = v_{Spt} - (v_2 - v_{Spt}) = 2v_{Spt} - v_2 \quad (46)$$

Oder mit den Formeln:

$$u_1 = \frac{m_1 - m_2}{m_1 + m_2} v_1 + \frac{2m_2}{m_1 + m_2} v_2 \quad (47)$$

$$u_2 = \frac{2m_1}{m_1 + m_2} v_1 + \frac{m_2 - m_1}{m_1 + m_2} v_2 \quad (48)$$

Vollständig inelastischer Stoß

Gegeben: Massen m_1 und m_2 , Anfangsgeschwindigkeiten v_1 und v_2

1. Gemeinsame Geschwindigkeit nach dem Stoß berechnen:

$$u_1 = u_2 = \frac{m_1 \cdot v_1 + m_2 \cdot v_2}{m_1 + m_2} \quad (49)$$

Elastischer Stoß mit ungleichen Massen

Wenn eine leichte Kugel auf eine ruhende schwere Kugel trifft ($m_1 < m_2$ und $v_2 = 0$):

$$u_1 = \frac{m_1 - m_2}{m_1 + m_2} v_1 \quad (50)$$

$$u_2 = \frac{2m_1}{m_1 + m_2} v_1 \quad (51)$$

Für $m_1 \ll m_2$ (z.B. Tennisball gegen Wand) gilt näherungsweise:

$$u_1 \approx -v_1 \quad (\text{Richtungsumkehr}) \quad (52)$$

$$u_2 \approx 0 \quad (\text{schwerer Körper bleibt fast in Ruhe}) \quad (53)$$

Umgekehrt, wenn eine schwere Kugel auf eine leichte, ruhende Kugel trifft ($m_1 > m_2$ und $v_2 = 0$):

$$u_1 \approx v_1 \quad (\text{fast unverändert}) \quad (54)$$

$$u_2 \approx 2v_1 \quad (\text{doppelte Geschwindigkeit des auftreffenden Körpers}) \quad (55)$$

Vergleich: Impuls- und Energieerhaltung

- Impulserhaltung gilt immer, wenn keine äußeren Kräfte wirken (Grundprinzip der Mechanik)
- Energieerhaltung gilt nur für konservative Vorgänge ohne Energieverluste
- Bei elastischen Stößen gelten beide Erhaltungssätze
- Bei inelastischen Stößen gilt nur die Impulserhaltung, die mechanische Energie nimmt ab
- Die Impulserhaltung liefert Vektorgleichungen (Richtungen wichtig)
- Die Energieerhaltung liefert eine Skalargleichung (nur Beträge wichtig)

Anwendungen der Impulserhaltung

Ballistisches Pendel

Ein ballistisches Pendel ist ein Gerät zur Messung der Geschwindigkeit von Projektilen. Es besteht aus einem Pendelkörper, in den das Projektil einschlägt und stecken bleibt (vollständig inelastischer Stoß).

Gegeben:

- Masse des Projektils m_K
- Masse des Pendels m_P
- Auslenkung des Pendels x
- Pendellänge L

Die Geschwindigkeit des Projektils lässt sich berechnen:

$$v_K = \frac{m_P + m_K}{m_K} \cdot \sqrt{2g \cdot (L - \sqrt{L^2 - x^2})} \quad (56)$$

Dabei wird ein zweistufiger Prozess betrachtet:

1. Inelastischer Stoß: Impulserhaltung, Energieverlust
2. Pendelschwingung: Energieerhaltung (kinetische zu potentieller Energie)

Raketenantrieb

Der Raketenantrieb basiert auf dem Rückstoßprinzip und der Impulserhaltung. Die Rakete stößt Treibstoff mit hoher Geschwindigkeit aus und erhält dadurch einen Impuls in die entgegengesetzte Richtung.

Die Raketengleichung (Ziolkowski-Gleichung) beschreibt die maximale Geschwindigkeit einer Rakete:

$$v_{\max} = v_{\text{rel}} \cdot \ln \frac{m_0}{m_{\text{leer}}} \quad (57)$$

Dabei ist:

- v_{rel} : Austrittsgeschwindigkeit des Treibstoffs relativ zur Rakete
- m_0 : Anfangsmasse (Rakete + Treibstoff)
- m_{leer} : Leermasse der Rakete (ohne Treibstoff)

Stoßsimulation in Unity

```
1 // Elastischen Stoss zwischen zwei Objekten simulieren
2 void OnCollisionEnter(Collision collision) {
3     Rigidbody otherRigidbody = collision.rigidbody;
4
5     // Geschwindigkeiten vor dem Stoss
6     Vector3 v1 = rigidbody.velocity;
7     Vector3 v2 = otherRigidbody.velocity;
8
9     // Massen
10    float m1 = rigidbody.mass;
11    float m2 = otherRigidbody.mass;
12
13    // Normale des Stosses (Verbindungsline der Mittelpunkte)
14    Vector3 normal = (otherRigidbody.position - rigidbody.position).normalized;
15
16    // Projektion der Geschwindigkeiten auf die Stossnormale
17    float v1n = Vector3.Dot(v1, normal);
18    float v2n = Vector3.Dot(v2, normal);
19
20    // Nur berechnen, wenn Objekte sich annaehern
21    if (v1n - v2n > 0) {
22        // Geschwindigkeiten nach dem Stoss entlang der Normalen
23        float u1n = ((m1 - m2) * v1n + 2 * m2 * v2n) / (m1 + m2);
24        float u2n = ((m2 - m1) * v2n + 2 * m1 * v1n) / (m1 + m2);
25
26        // Geschwindigkeitsaenderung nur entlang der Normalen
27        Vector3 v1Tangential = v1 - v1n * normal;
28        Vector3 v2Tangential = v2 - v2n * normal;
29
30        // Neue Geschwindigkeiten
31        Vector3 newV1 = v1Tangential + u1n * normal;
32        Vector3 newV2 = v2Tangential + u2n * normal;
33
34        // Geschwindigkeiten setzen
35        rigidbody.velocity = newV1;
36        otherRigidbody.velocity = newV2;
37    }
38 }
```

Arbeit und Energie

Arbeit

Arbeit

Die physikalische Arbeit W ist definiert als das Skalarprodukt aus Kraft und Weg:

$$W = \int_{\vec{r}_0}^{\vec{r}_1} \vec{F} \cdot d\vec{r} \tag{58}$$

Die Einheit der Arbeit ist Joule (J): $1 \text{ J} = 1 \text{ N} \cdot \text{m} = 1 \frac{\text{kg} \cdot \text{m}^2}{\text{s}^2}$

Für den Spezialfall einer konstanten Kraft in Richtung des Weges vereinfacht sich die Formel zu:

$$W = F \cdot \Delta x \tag{59}$$

Arbeit mit Winkel zwischen Kraft und Weg

Wenn die Kraft einen Winkel α mit der Wegrichtung einschließt, gilt:

$$W = F \cdot \cos(\alpha) \cdot \Delta x \tag{60}$$

Nur die Komponente der Kraft in Richtung des Weges verrichtet Arbeit.

Graphische Darstellung der Arbeit

In einem Weg-Kraft-Diagramm entspricht die Arbeit der Fläche unter der Kraft-Kurve:

$$W = \int_{x_0}^{x_1} F_x(x) dx \tag{61}$$

Beispiele:

- Konstante Kraft: $W = F \cdot \Delta x$ (Rechteck)
- Linear ansteigende Kraft: $W = \frac{1}{2} \cdot F_{max} \cdot \Delta x$ (Dreieck)

Formen physikalischer Arbeit

Arten von Arbeit

Verschiedene Arten physikalischer Arbeit:

- Hubarbeit: $W = m \cdot g \cdot h$
- Beschleunigungsarbeit: $W = \frac{1}{2} \cdot m \cdot (v_1^2 - v_0^2)$
- Deformationsarbeit (Federspannung): $W = \frac{1}{2} \cdot k \cdot (x_1^2 - x_0^2)$
- Reibungsarbeit: $W = \mu \cdot F_N \cdot \Delta x$

Berechnung der Arbeit in Praxisbeispielen

Hubarbeit

- Gegeben: Masse m , Höhendifferenz h
- Berechnung: $W = m \cdot g \cdot h$
- Beispiel: Ein 20 kg schwerer Körper wird 5 m angehoben: $W = 20 \text{ kg} \cdot 9.81 \frac{\text{m}}{\text{s}^2} \cdot 5 \text{ m} = 981 \text{ J}$

Federarbeit

- Gegeben: Federkonstante k , Auslenkungen x_0 und x_1
- Berechnung: $W = \frac{1}{2} \cdot k \cdot (x_1^2 - x_0^2)$
- Beispiel: Eine Feder mit $k = 100 \text{ N/m}$ wird von 0 auf 0.2 m gedehnt: $W = \frac{1}{2} \cdot 100 \frac{\text{N}}{\text{m}} \cdot (0.2^2 - 0^2) \text{ m}^2 = 2 \text{ J}$

Reibungsarbeit

- Gegeben: Reibungskoeffizient μ , Normalkraft F_N , Strecke Δx
- Berechnung: $W = \mu \cdot F_N \cdot \Delta x$
- Beispiel: Ein 5 kg schwerer Körper wird 10 m weit gezogen mit $\mu = 0.2$: $W = 0.2 \cdot (5 \text{ kg} \cdot 9.81 \frac{\text{m}}{\text{s}^2}) \cdot 10 \text{ m} = 98.1 \text{ J}$

Energie

Energie

Energie ist die Fähigkeit eines Systems, Arbeit zu verrichten. Die an einem Körper verrichtete Arbeit ist gleich der Änderung seiner Energie:

$$W = \Delta E = E_{nachher} - E_{vorher} \tag{62}$$

Die Einheit der Energie ist ebenfalls Joule (J).

Während Arbeit einen Prozess beschreibt (Prozessgröße), charakterisiert Energie einen Zustand (Zustandsgröße).

Energieformen

Die wichtigsten Energieformen in der Mechanik:

- Kinetische Energie (Bewegungsenergie): $E_{kin} = \frac{1}{2} \cdot m \cdot v^2$
- Potentielle Energie im Schwerfeld: $E_{pot} = m \cdot g \cdot h$
- Spannenergie einer Feder: $E_{spann} = \frac{1}{2} \cdot k \cdot x^2$
- Rotationsenergie: $E_{rot} = \frac{1}{2} \cdot J \cdot \omega^2$

Energieerhaltung

Das Prinzip der Energieerhaltung besagt, dass in einem abgeschlossenen System ohne Einwirkung nicht-konservativer Kräfte die Gesamtenergie konstant bleibt:

$$E_{ges} = E_{kin} + E_{pot} + E_{spann} + \dots = \text{const.} \tag{63}$$

Bei konservativen Kräften bleibt die mechanische Energie erhalten. Bei dissipativen Kräften (wie Reibung) wird mechanische Energie in andere Energieformen (meist Wärme) umgewandelt.

Energiewandlungen beim Pendel

Ein Pendel der Masse m und Länge l zeigt deutlich die Umwandlung zwischen potentieller und kinetischer Energie:

- Am höchsten Punkt: maximale potentielle Energie, keine kinetische Energie

$$E_{pot,max} = m \cdot g \cdot h = m \cdot g \cdot l \cdot (1 - \cos \alpha) \tag{64}$$

- Am tiefsten Punkt: maximale kinetische Energie, minimale potentielle Energie

$$E_{kin,max} = \frac{1}{2} \cdot m \cdot v_{max}^2 = m \cdot g \cdot l \cdot (1 - \cos \alpha) \tag{65}$$

- Die Gesamtenergie $E_{ges} = E_{pot} + E_{kin}$ bleibt konstant (bei Vernachlässigung der Reibung)

Anwendung der Energieerhaltung zur Problemlösung

Allgemeines Vorgehen

1. Identifiziere die relevanten Energieformen im Anfangs- und Endzustand
2. Stelle die Energieerhaltungsgleichung auf (bei konservativen Systemen)
3. Berücksichtige eventuelle Energieverluste durch Reibung
4. Löse die Gleichung nach der gesuchten Größe auf

Beispiel: Ball wird von Höhe h_1 fallen gelassen und springt auf Höhe h_2

- Anfangszustand: $E_1 = m \cdot g \cdot h_1$
- Endzustand: $E_2 = m \cdot g \cdot h_2$
- Bei teilelastischem Stoß gilt: $h_2 = e^2 \cdot h_1$, wobei e der Restitutionskoeffizient ist
- Der Energieverlust beträgt: $\Delta E = m \cdot g \cdot (h_1 - h_2) = m \cdot g \cdot h_1 \cdot (1 - e^2)$

Leistung und Wirkungsgrad

Leistung

Die Leistung P ist die pro Zeiteinheit verrichtete Arbeit:

$$P = \frac{dW}{dt}$$

(66)

Die Einheit der Leistung ist Watt (W): $1\text{ W} = 1\frac{\text{J}}{\text{s}} = 1\frac{\text{kg}\cdot\text{m}^2}{\text{s}^3}$
Für konstante Leistung gilt:

$$W = P \cdot t$$

(67)

Wirkungsgrad

Der Wirkungsgrad η ist das Verhältnis von Nutzenergie zu zugeführter Energie:

$$\eta = \frac{E_{Nutz}}{E_{zugef\ddot{u}hrt}}$$

(68)

Der Wirkungsgrad ist eine dimensionslose Zahl zwischen 0 und 1 (oder in Prozent ausgedrückt: 0-100%).

Leistung und Wirkungsgrad einer Pumpe

Eine Pumpe hebt Wasser aus einer Tiefe von 5 m auf eine Höhe von 15 m. Sie liefert 0.2 m³ Wasser pro Minute und benötigt eine elektrische Leistung von 700 W.

- Nützliche Leistung:

$$P_{Nutz} = \frac{\Delta E_{pot}}{\Delta t} = \frac{m \cdot g \cdot \Delta h}{\Delta t}$$

(69)

$$= \rho \cdot V \cdot g \cdot \Delta h / \Delta t$$

(70)

$$= 1000 \frac{\text{kg}}{\text{m}^3} \cdot 0.2 \frac{\text{m}^3}{60 \text{ s}} \cdot 9.81 \frac{\text{m}}{\text{s}^2} \cdot 20 \text{ m}$$

(71)

$$\approx 654 \text{ W}$$

(72)

- Wirkungsgrad:

$$\eta = \frac{P_{Nutz}}{P_{el}} = \frac{654 \text{ W}}{700 \text{ W}} \approx 0.93 = 93\%$$

(73)

Energie und Leistung in Unity

```
1 // Berechnung und Ueberwachung der Energien in Unity
2 void CalculateEnergies() {
3     // Kinetische Energie
4     float kineticEnergy = 0.5f * rigidbody.mass *
5         rigidbody.velocity.sqrMagnitude;
6
7     // Potentielle Energie (Hoehe relativ zum Boden)
8     float potentialEnergy = rigidbody.mass * 9.81f *
9         (transform.position.y - groundLevel);
10
11     // Spannenergie (falls zutreffend)
12     float springEnergy = 0.5f * springConstant *
13         (transform.position.x - equilibriumPosition)^2;
14
15     // Gesamtenergie
16     float totalEnergy = kineticEnergy + potentialEnergy + springEnergy;
17
18     // Ueberwachen der Energieerhaltung
19     if (Mathf.Abs(totalEnergy - initialTotalEnergy) > tolerance) {
20         Debug.LogWarning("Energy not conserved! Check for numerical errors or
21             non-conservative forces.");
22     }
23
24     // Momentane Leistung berechnen (Energieaenderung pro Zeit)
25     float power = (totalEnergy - lastTotalEnergy) / Time.deltaTime;
26     lastTotalEnergy = totalEnergy;
27 }
```

Rotation

Kinematik der Rotation

Rotationsbewegung

Eine Rotationsbewegung ist die Drehung eines Körpers um eine bestimmte Achse. Die Beschreibung erfolgt analog zur Translation, jedoch werden andere Größen verwendet:

- Drehwinkel φ statt Ort \vec{r}
- Winkelgeschwindigkeit ω statt Geschwindigkeit \vec{v}
- Winkelbeschleunigung α statt Beschleunigung \vec{a}

Freiheitsgrade

Ein starrer Körper im dreidimensionalen Raum hat sechs Freiheitsgrade:

- Drei translatorische Freiheitsgrade (Bewegung in x-, y- und z-Richtung)
- Drei rotatorische Freiheitsgrade (Drehung um die x-, y- und z-Achse)

Diese Bewegungen sind für einen freien Körper unabhängig voneinander.

Zusammenhänge der rotatorischen Größen

Analog zur Translation gelten folgende Beziehungen:

$$\omega = \frac{d\varphi}{dt} \tag{74}$$

$$\alpha = \frac{d\omega}{dt} \tag{75}$$

$$\varphi = \int \omega \, dt \tag{76}$$

$$\omega = \int \alpha \, dt \tag{77}$$

Die Maßeinheiten sind:

- Winkel φ : Radian (rad), dimensionslos
- Winkelgeschwindigkeit ω : rad/s
- Winkelbeschleunigung α : rad/s²

Winkel in Radian

Der Winkel in Radian ist definiert als das Verhältnis von Kreisbogen s zu Radius r :

$$\varphi = \frac{s}{r} \tag{78}$$

Umrechnung zwischen Grad und Radian:

$$\varphi_{rad} = \varphi_{deg} \cdot \frac{\pi}{180} \tag{79}$$

Wichtige Werte:

- Vollwinkel: $360^\circ = 2\pi$ rad
- Rechter Winkel: $90^\circ = \frac{\pi}{2}$ rad

Zusammenhang zwischen Bahngeschwindigkeit und Winkelgeschwindigkeit

Bei einer Kreisbewegung mit Radius r gilt für die Bahngeschwindigkeit v :

$$v = \omega \cdot r \tag{80}$$

Dabei ist v die Geschwindigkeit eines Punktes auf dem Umfang des Kreises (tangential zur Kreisbahn) und ω die Winkelgeschwindigkeit der Rotation.

Schwerpunkt und Trägheitsmoment

Schwerpunkt

Der Schwerpunkt (auch Massenmittelpunkt genannt) eines Körpers ist der gewichtete Mittelwert der Positionen aller Massenelemente:

$$\vec{r}_S = \frac{1}{\sum_{i=1}^n m_i} \cdot \sum_{i=1}^n m_i \cdot \vec{r}_i \tag{81}$$

Für kontinuierliche Massenverteilungen:

$$\vec{r}_S = \frac{1}{M} \iiint_K \vec{r} \rho(\vec{r}) \, dV \tag{82}$$

wobei $\rho(\vec{r})$ die Dichteverteilung und M die Gesamtmasse ist.

Schwerpunktsatz

Der Schwerpunktsatz besagt, dass eine an einem beliebigen Punkt eines starren Körpers angreifende Kraft

- eine Translation des Schwerpunktes bewirkt, so als ob die Kraft direkt am Schwerpunkt angreifen würde, und
- ein Drehmoment um den Schwerpunkt erzeugt, wenn die Kraftwirkungslinie nicht durch den Schwerpunkt verläuft.

Dies erlaubt die Zerlegung der Bewegung in eine Translation des Schwerpunktes und eine Rotation um den Schwerpunkt.

Trägheitsmoment

Das Trägheitsmoment J ist ein Maß für den Widerstand eines Körpers gegenüber Rotationsbeschleunigungen.

Es ist definiert als:

$$J = \sum_{i=1}^n m_i \cdot r_i^2 \tag{83}$$

Für kontinuierliche Massenverteilungen:

$$J = \iiint_K r^2 \rho(\vec{r}) \, dV \tag{84}$$

Dabei ist r der senkrechte Abstand des Massenelements von der Drehachse.

Trägheitsmomente einfacher Körper

Einige wichtige Trägheitsmomente:

- Punktmasse m im Abstand r von der Drehachse: $J = m \cdot r^2$
- Dünner Stab der Länge L und Masse m (Drehachse durch Mitte, senkrecht zum Stab): $J = \frac{1}{12} \cdot m \cdot L^2$
- Dünner Stab (Drehachse am Ende): $J = \frac{1}{3} \cdot m \cdot L^2$
- Vollzylinder der Masse m und Radius R (Drehachse = Symmetrieachse): $J = \frac{1}{2} \cdot m \cdot R^2$
- Hohlzylinder (Drehachse = Symmetrieachse): $J = m \cdot R^2$
- Kugel der Masse m und Radius R (Drehachse durch Mittelpunkt): $J = \frac{2}{5} \cdot m \cdot R^2$

Steiner'scher Satz

Der Steiner'sche Satz beschreibt, wie sich das Trägheitsmoment ändert, wenn die Drehachse parallel verschoben wird:

$$J = J_S + m \cdot d^2 \tag{85}$$

Dabei ist:

- J_S das Trägheitsmoment bezüglich einer Achse durch den Schwerpunkt
- m die Gesamtmasse des Körpers
- d der Abstand zwischen den parallelen Achsen

Drehmoment

Das Drehmoment \vec{M} ist die Ursache einer Rotationsbeschleunigung. Es ist definiert als das Kreuzprodukt aus Hebelarm \vec{r} und Kraft \vec{F} :

$$\vec{M} = \vec{r} \times \vec{F}$$

(86)

Der Betrag des Drehmoments ist:

$$M = r \cdot F \cdot \sin(\alpha)$$

(87)

wobei α der Winkel zwischen Hebelarm und Kraft ist.

Alternativ:

$$M = F \cdot d$$

(88)

wobei d der senkrechte Abstand der Kraftwirkungslinie von der Drehachse ist.

Zweites Newton'sches Gesetz für Rotationen

Analog zum zweiten Newton'schen Gesetz für translatorische Bewegungen gilt für Rotationen:

$$\vec{M} = J \cdot \vec{\alpha}$$

(89)

Diese Gleichung verknüpft das Drehmoment mit der resultierenden Winkelbeschleunigung.

Drehimpuls

Der Drehimpuls \vec{L} ist das rotatorische Analogon zum linearen Impuls. Er ist definiert als:

$$\vec{L} = J \cdot \vec{\omega}$$

(90)

Für einen Punktteilchen mit Impuls \vec{p} im Abstand \vec{r} von der Drehachse:

$$\vec{L} = \vec{r} \times \vec{p}$$

(91)

Das Drehmoment ändert den Drehimpuls gemäß:

$$\vec{M} = \frac{d\vec{L}}{dt}$$

(92)

Drehimpulserhaltung

In Abwesenheit äußerer Drehmomente bleibt der Drehimpuls eines Systems konstant:

$$\vec{L} = \text{const.}$$

(93)

Die Drehimpulserhaltung ist neben der Impuls- und Energieerhaltung ein fundamentales Erhaltungsgesetz der Physik.

Beispiele:

- Eiskunstläufer, der die Arme anzieht, um schneller zu rotieren
- Stabilität von Fahrrädern aufgrund der rotierenden Räder
- Präzession eines Kreiseis

Rotationsenergie

Die kinetische Energie der Rotation (Rotationsenergie) ist gegeben durch:

$$E_{rot} = \frac{1}{2} \cdot J \cdot \omega^2$$

(94)

Die Gesamtenergie eines Körpers, der sowohl translatorische als auch rotatorische Bewegung erfährt, ist:

$$E_{ges} = E_{trans} + E_{rot} = \frac{1}{2} \cdot m \cdot v^2 + \frac{1}{2} \cdot J \cdot \omega^2$$

(95)

Berechnung rotatorischer Größen

Berechnung des Schwerpunkts zusammengesetzter Körper

1. Teile den Körper in einfache Formen mit bekannten Schwerpunkten
2. Berechne die Gesamtmasse: $M = \sum m_i$
3. Berechne den Schwerpunkt: $\vec{r}_S = \frac{1}{M} \sum m_i \cdot \vec{r}_i$

Berechnung des Drehmoments

1. Identifiziere alle wirkenden Kräfte \vec{F}_i
2. Bestimme ihre Hebelarme \vec{r}_i bezüglich der Drehachse
3. Berechne die einzelnen Drehmomente: $\vec{M}_i = \vec{r}_i \times \vec{F}_i$
4. Summiere alle Drehmomente: $\vec{M}_{ges} = \sum \vec{M}_i$

Winkelbeschleunigung aus Drehmoment

1. Berechne das resultierende Drehmoment \vec{M}_{ges}
2. Bestimme das Trägheitsmoment J bezüglich der Drehachse
3. Berechne die Winkelbeschleunigung: $\vec{\alpha} = \frac{\vec{M}_{ges}}{J}$

Drehmoment in Unity

```
1 // Drehmoment auf einen Koerper aufbringen
2 void ApplyTorque() {
3     // Angriffspunkt relativ zum Massenmittelpunkt des Koerpers
4     Vector3 relativePos = transform.TransformPoint(localForcePoint) -
5                           rigidbody.worldCenterOfMass;
6
7     // Kraft berechnen (z.B. in diesem Fall eine konstante Kraft)
8     Vector3 force = new Vector3(0, 0, forceAmount);
9
10    // Drehmoment als Kreuzprodukt aus Hebelarm und Kraft
11    Vector3 torque = Vector3.Cross(relativePos, force);
12
13    // Kraft und Drehmoment auf den Koerper anwenden
14    rigidbody.AddForceAtPosition(force, transform.TransformPoint(localForcePoint));
15
16    // Alternative: Drehmoment direkt anwenden
17    // rigidbody.AddTorque(torque);
18 }
```

Rotation eines Würfels über eine Kante

Ein Würfel der Kantenlänge a und Masse m wird über seine Kante gestoßen.

- Trägheitsmoment bezüglich der Kante: $J = \frac{m}{6}(2a^2) = \frac{m \cdot a^2}{3}$
- Wenn der Schwerpunkt über die Kante hinausragt, wirkt ein Drehmoment aufgrund der Gewichtskraft:
 $M = m \cdot g \cdot \frac{a}{2} \cdot \sin \varphi$ wobei φ der Winkel zwischen der Vertikalen und der Linie vom Drehpunkt zum Schwerpunkt ist.
- Die resultierende Winkelbeschleunigung ist: $\alpha = \frac{M}{J} = \frac{3 \cdot g \cdot \sin \varphi}{2a}$
- Beim horizontalen Stoßen mit einer Kraft F entlang einer Linie, die um h vom Schwerpunkt versetzt ist, entsteht ein Drehmoment: $M = F \cdot h$
- Dies führt zu einer kombinierten Translation und Rotation des Würfels.

Euler-Winkel

Euler-Winkel sind ein Satz von drei Winkeln, die eine Rotation im dreidimensionalen Raum vollständig beschreiben:

- Pitch (φ): Drehung um die x-Achse
- Yaw (θ): Drehung um die y-Achse
- Roll (ψ): Drehung um die z-Achse

Bei der Anwendung von Euler-Winkeln ist die Reihenfolge der Drehungen entscheidend. Außerdem muss unterschieden werden zwischen:

- Intrinsischen Drehungen: Die Achsen drehen sich mit dem Objekt
- Extrinsischen Drehungen: Die Achsen bleiben fixiert

Quaternionen in Unity

Unity verwendet intern Quaternionen zur Darstellung von Rotationen, da diese mehrere Vorteile gegenüber Euler-Winkeln bieten:

- Keine "Gimbal-LockProbleme (Verlust eines Freiheitsgrades bei bestimmten Orientierungen)
- Effizientere Interpolation zwischen Rotationen
- Numerisch stabiler

Quaternionen haben die Form $q = a + bi + cj + dk$, wobei $i^2 = j^2 = k^2 = ijk = -1$. Eine Rotation um eine Achse \vec{n} mit dem Winkel α wird als Quaternion dargestellt:

$$q = \cos\left(\frac{\alpha}{2}\right) + \sin\left(\frac{\alpha}{2}\right)(n_x i + n_y j + n_z k)$$

(96)

Arbeiten mit Quaternionen in Unity

Erzeugung einer Quaternion

```
1 // Eine Quaternion aus Euler-Winkeln erzeugen
2 Quaternion rotation = Quaternion.Euler(30f, 45f, 0f); // Drehung: 30 Grad um x, 45
  Grad um y
3
4 // Eine Quaternion aus einer Achse und einem Winkel erzeugen
5 Vector3 axis = new Vector3(0, 1, 0); // y-Achse
6 float angle = 45f; // 45 Grad
7 Quaternion fromAxisAngle = Quaternion.AngleAxis(angle, axis);
```

Anwenden einer Rotation

```
1 // Rotationen kombinieren (zuerst rotA, dann rotB)
2 Quaternion combinedRotation = rotB * rotA;
3
4 // Quaternion auf einen Vektor anwenden, um ihn zu rotieren
5 Vector3 rotatedVector = myQuaternion * originalVector;
6
7 // Objekt rotieren
8 transform.rotation = myQuaternion;
9
10 // Inkrementelle Rotation
11 transform.rotation = myQuaternion * transform.rotation;
```

Interpolation zwischen Rotationen

```
1 // Sphaerische Interpolation zwischen zwei Rotationen
2 Quaternion interpolated = Quaternion.Slerp(startRotation, endRotation, t);
3 // t ist ein Wert zwischen 0 und 1
```

Standfestigkeit

Ein Körper ist standfest, wenn die Wirkungslinie der Gewichtskraft durch die Unterstützungsfläche verläuft. Man unterscheidet drei Arten von Gleichgewicht:

- Stabiles Gleichgewicht: Nach einer kleinen Auslenkung kehrt der Körper in seine Ausgangslage zurück (potentielle Energie im Minimum)
- Labiles Gleichgewicht: Nach einer kleinen Auslenkung entfernt sich der Körper weiter von seiner Ausgangslage (potentielle Energie im Maximum)
- Indifferentes Gleichgewicht: Nach einer Auslenkung verharrt der Körper in der neuen Position (potentielle Energie konstant)

Kippkriterium

Ein Körper kippt, wenn die Wirkungslinie der Gewichtskraft außerhalb der Unterstützungsfläche verläuft. Für einen Quader mit Grundfläche $b \times l$ und Höhe h auf einer um den Winkel α geneigten Ebene gilt als Kippkriterium:

$$\tan \alpha > \frac{b}{2h}$$

(97)

Dabei ist b die Breite des Quaders in Richtung der Neigung.

Standfestigkeit eines Turms

Ein Turm aus mehreren Bausteinen ist standfest, wenn der Schwerpunkt jedes Teilstücks (bestehend aus allen Steinen oberhalb eines bestimmten Punktes) über der Unterstützungsfläche liegt. Bei einem Überhang von Bausteinen muss der Schwerpunkt der überhängenden Steine durch das Gegengewicht der nicht überhängenden Steine ausgeglichen werden, damit der gemeinsame Schwerpunkt noch über der Unterstützungsfläche bleibt. Der maximale Überhang für n identische Bausteine, die jeweils um einen bestimmten Abstand verschoben sind, wird durch die harmonische Reihe beschrieben:

$$\text{Überhang}_{\max} = \frac{L}{2} \cdot \sum_{i=1}^n \frac{1}{i}$$

(98)

wobei L die Länge eines Bausteins ist.

In Unity ist die korrekte Modellierung von Standfestigkeit und Gleichgewicht wichtig für:

- Realistische Darstellung von stehenden oder gestapelten Objekten
- Simulierte Reaktionen auf Stöße und Kräfte
- Korrekte Kippsimulationen und Fallbewegungen

Um zuverlässige Ergebnisse zu erhalten, müssen sowohl die Kollisionsgeometrie als auch die Massenverteilung (insbesondere die Position des Schwerpunkts) korrekt definiert werden.