Physic Engines

Jil Zerndt
FS 2025

Einführung und Grundlagen

Physics Engines Physics Engines sind Softwarekomponenten, die physikalische Effekte in Computerprogrammen simulieren. Unity verwendet PhysX als Standard-Physics-Engine.

Ziele des Moduls:
- Physikalische Modellierung in Unity verstehen
- Grundprinzipien der Mechanik für realistische Simulationen anwenden
- Kopplung von Physiksimulatoren mit realistischen Parametern

Modellbildungsprozess: Wirklichkeit → Physikalisches Modell → Mathematisches Modell → Numerisches Modell → Unity-Implementation

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):

$$\vec{r}_1 \times \vec{r}_2 = \begin{pmatrix} r_{y1}r_{z2} - r_{z1}r_{y2} \\ r_{z1}r_{x2} - r_{x1}r_{z2} \\ r_{x1}r_{y2} - r_{y1}r_{x2} \end{pmatrix}$$

SI-Einheiten
- 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.

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

Unity-Physik Grundlagen **Wichtige Konzepte:**
- Physikalische Größen immer mit Einheiten kommentieren
- Rigidbody für Position/Geschwindigkeit verwenden
- AddForce() für Kraftanwendung
- FixedUpdate() für Physikberechnungen

Vector3 in Unity
- `Vector3` für 3D-Positionen und -Richtungen
- Wichtige Eigenschaften: `Vector3.forward`, `Vector3.up`, `Vector3.right`
- Operationen: Skalarprodukt, Kreuzprodukt, Betrag, Normalisierung

Unity vs. Standard-Koordinatensystem
Unterschiede:
- Standard-Physik: Rechtssystem
- Unity: Linkssystem
- Unity: positive y-Achse zeigt nach oben

Auswirkungen: Unterschiedliche Kreuzprodukt-Ergebnisse und Rotationsrichtungen.

Kinematik

Kinematik Die Kinematik beschreibt die Bewegung ohne Betrachtung der Ursachen. Eine Bewegung wird vollständig charakterisiert durch:
- Ort: $\vec{r}(t)$
- Geschwindigkeit: $\vec{v}(t) = \frac{d\vec{r}}{dt}$
- Beschleunigung: $\vec{a}(t) = \frac{d\vec{v}}{dt} = \frac{d^2\vec{r}}{dt^2}$

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: $\vec{v} = \int \vec{a}\, dt$

Geschwindigkeit und Beschleunigung

Mittlere vs. Momentangeschwindigkeit
Mittlere Geschwindigkeit:

$$\bar{v}_x = \frac{\Delta r_x}{\Delta t} = \frac{r_x(t_2) - r_x(t_1)}{t_2 - t_1}$$

Momentangeschwindigkeit:

$$v_x(t) = \lim_{\Delta t \to 0} \frac{\Delta r_x}{\Delta t} = \frac{dr_x}{dt}$$

Mittlere vs. Momentanbeschleunigung
Mittlere Beschleunigung:

$$\bar{a}_x = \frac{\Delta v_x}{\Delta t} = \frac{v_x(t_2) - v_x(t_1)}{t_2 - t_1}$$

Momentanbeschleunigung:

$$a_x(t) = \lim_{\Delta t \to 0} \frac{\Delta v_x}{\Delta t} = \frac{dv_x}{dt} = \frac{d^2 r_x}{dt^2}$$

Unterschied Geschwindigkeit und Schnelligkeit:
- Geschwindigkeit: Vektorielle Größe mit Richtung
- Schnelligkeit: Betrag der Geschwindigkeit (Skalar)
- $|\vec{v}| = \sqrt{v_x^2 + v_y^2 + v_z^2}$

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

Momentangeschwindigkeit und -beschleunigung

Momentangeschwindigkeit Die Momentangeschwindigkeit zur Zeit t_0 ist definiert als:

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

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

Der Betrag der Geschwindigkeit wird oft als Schnelligkeit bezeichnet:

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

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:

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

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: $\frac{d}{dx}(\ln x) = \frac{1}{x}$

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: $\frac{d}{dt}(f(g(t))) = \frac{df}{dg} \cdot \frac{dg}{dt}$

Sinus/Kosinus $\frac{d}{dx}(\sin x) = \cos x$, $\frac{d}{dx}(\cos x) = -\sin x$

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: $v(t) = v_0 + at$
2. Position in Abhängigkeit von der Zeit bestimmen: $r(t) = r_0 + v_0 t + \frac{1}{2}at^2$
3. Alternative Formel bei bekannter Strecke (ohne Zeit): $v^2 = v_0^2 + 2a(r - r_0)$

Freier Fall
Ein Körper fällt aus der Höhe r_0 mit Anfangsgeschwindigkeit $v_0 = 0$.
- Beschleunigung: $a(t) = -g$ (g = 9.81 m/s²)
- 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: $t_{gesamt} = \frac{2v_0}{g}$

Unity-Implementation

Unity Position/Geschwindigkeit
Grundoperationen:
- transform.position für Ortsvektor
- Verschiebung als Vektordifferenz
- Position mit velocity * Time.deltaTime aktualisieren

Kinematik-Script Struktur
Wichtige Komponenten:
- Anfangswerte (Position, Geschwindigkeit) speichern
- Zeitvariable t = Time.time - startTime
- Kinematische Gleichung in Update() anwenden
- transform.position direkt setzen

Spezialfälle der Bewegung
Gleichförmige Bewegung ($\vec{a} = 0$):

$$\vec{r}(t) = \vec{r}_0 + \vec{v}t$$

Freier Fall in Unity ($\vec{a} = -g\hat{j}$):
- $g = 9.81\,m/s^2$ (Erdbeschleunigung)
- Unity Standard-Gravitation: $-9.81\,m/s^2$ in Y-Richtung

Unity Kinematik
Implementierung konstanter Beschleunigung:
- Geschwindigkeit: $v = v_0 + at$ mit Time.deltaTime
- Position direkt: $r = r_0 + v_0 t + \frac{1}{2}at^2$
- FixedUpdate() für Physikberechnungen verwenden

Kinematische Probleme lösen
Schritt 1: Bekannte Variablen identifizieren
- Anfangsposition $\vec{r}_0$
- Anfangsgeschwindigkeit $\vec{v}_0$
- Beschleunigung $\vec{a}$
- Zeit t oder End-Position/Geschwindigkeit

Schritt 2: Passende Gleichung wählen
- $\vec{v}(t) = \vec{v}_0 + \vec{a}t$ wenn Zeit bekannt ist
- $\vec{r}(t) = \vec{r}_0 + \vec{v}_0 t + \frac{1}{2}\vec{a}t^2$ für Position
- $\vec{v}^2 = \vec{v}_0^2 + 2\vec{a} \cdot \Delta\vec{r}$ wenn Zeit unbekannt ist

Schritt 3: Komponentenweise lösen
- Vektoren in x-, y-, z-Komponenten aufteilen
- Jede Komponente unabhängig lösen
- Ergebnisse zum finalen Vektor kombinieren

Wurfbewegung: Ein Ball wird aus Höhe $h = 10m$ mit Anfangsgeschwindigkeit $\vec{v}_0 = (5, 8, 0)\,m/s$ geworfen. Berechne Zeit bis zum Aufprall und horizontale Distanz.
Gegeben: $\vec{r}_0 = (0, 10, 0)$, $\vec{v}_0 = (5, 8, 0)$, $\vec{a} = (0, -9.81, 0)$
Y-Komponente (vertikal): $y(t) = 10 + 8t - 4.905t^2$
Ball trifft Boden bei $y(t) = 0$: $10 + 8t - 4.905t^2 = 0$ $t = 2.24s$ (mit quadratischer Formel)
X-Komponente (horizontal): $x(t) = 5t = 5 \times 2.24 = 11.2m$

Dynamik

Dynamik Die Dynamik untersucht die Ursachen von Bewegungen - die wirkenden Kräfte. Grundlage sind die Newton'schen Gesetze.

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.

Newton'sche Gesetze

Trägheitsgesetz
Das erste Newton'sche Gesetz:

$$\vec{F} = 0 \Rightarrow \vec{v} = \text{const.}$$

Dies bedeutet auch: $\vec{F} = 0 \Rightarrow \vec{a} = 0$
Das Gesetz gilt nur in Inertialsystemen (Bezugssysteme ohne Beschleunigung oder Rotation).

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

$$\vec{p} = m \cdot \vec{v}$$

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

Bewegungsgesetz
Das zweite Newton'sche Gesetz in seiner allgemeinen Form:

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

Für Körper mit konstanter Masse:

$$\vec{F} = m \cdot \vec{a}$$

In integraler Form:

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

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

Wechselwirkungsgesetz
Das dritte Newton'sche Gesetz:

$$\vec{F}_{12} = -\vec{F}_{21}$$

Kräftepaare haben folgende Eigenschaften:
1. Gleicher Betrag, entgegengesetzte Richtung
2. Greifen an verschiedenen Körpern an
3. Haben die gleiche physikalische Ursache

Superpositionsprinzip
Mehrere Kräfte addieren sich vektoriell:

$$\vec{F}_{res} = \sum_{i=1}^{n} \vec{F}_i$$

Für jede Komponente:

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

Grundlegende Kräfte
Gravitationskraft (nahe Erdoberfläche):
$\vec{F}_g = m\vec{g}$ mit $g = 9.81\,m/s^2$
Federkraft (Hooke'sches Gesetz): $\vec{F}_s = -k\Delta\vec{l}$
Reibungskraft:
- Haftreibung: $f_s \leq \mu_s N$
- Gleitreibung: $f_k = \mu_k N$

Dämpfungskraft:
$\vec{F}_d = -b\vec{v}$ (geschwindigkeitsproportional)

Harmonischer Oszillator

Harmonische Schwingung
Bei einer Federkraft entsteht eine harmonische Schwingung mit der Bewegungsgleichung:

$$m\frac{d^2x}{dt^2} = -kx$$

Lösung: $x(t) = A\cos(\omega t + \phi)$
wobei $\omega = \sqrt{\frac{k}{m}}$ die Kreisfrequenz ist.

Eigenschaften des harmonischen Oszillators
Kreisfrequenz: $\omega = \sqrt{\frac{k}{m}}$
Schwingungsdauer: $T = \frac{2\pi}{\omega} = 2\pi\sqrt{\frac{m}{k}}$
Frequenz: $f = \frac{1}{T} = \frac{\omega}{2\pi}$
Gesamtenergie: $E = \frac{1}{2}kA^2$ (konstant)

Unity Implementation

Unity Force Modes Unity bietet verschiedene Modi für Kraftanwendung über `Rigidbody.AddForce()`:
- `Force`: Kontinuierliche Kraft mit Masse (Standard)
- `Acceleration`: Kontinuierliche Kraft ohne Masse
- `Impulse`: Impulsive Kraft mit Masse
- `VelocityChange`: Impulsive Kraft ohne Masse

Unity Kraftanwendung **Wichtige Konzepte:**
- Rigidbody.AddForce() für Kräfte
- Federkraft: -k * position
- Dämpfung: -b * velocity
- FixedUpdate() für stabile Physik

Harmonischer Oszillator Unity **Implementierung:**
- $\omega = \sqrt{k/m}$ berechnen
- Position: $x = A\cos(\omega t + \phi)$
- Transform.position direkt setzen
- Equilibrium-Position als Referenz

Drehmoment

Drehmoment Rotatorisches Äquivalent zur Kraft:

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

Für Rotation um feste Achse: $\tau = rF\sin\theta$

Unity Drehmoment **Anwendung:**
- Rigidbody.AddTorque() für Rotation
- Vector3.up/right/forward für Achsen
- Rotationsfeder mit eulerAngles
- Input-gesteuerte Drehung

Problemlösungsstrategie für Kräfte

Schritt 1: System identifizieren
- Untersuchungsobjekt(e) definieren
- Koordinatensystem wählen
- Zeitintervall bestimmen

Schritt 2: Freikörperdiagramm zeichnen
- Alle auf das Objekt wirkenden Kräfte zeigen
- Kraftvektoren mit Symbolen beschriften
- Keine Kräfte einbeziehen, die das Objekt auf andere ausübt

Schritt 3: Newton'sches Bewegungsgesetz anwenden
- $\sum \vec{F} = m\vec{a}$ in Komponentenform schreiben
- $\sum F_x = ma_x$, $\sum F_y = ma_y$, $\sum F_z = ma_z$
- Unbekannte Größen lösen

Schritt 4: In Unity implementieren
- `Rigidbody.AddForce()` für jede Kraft verwenden
- Passenden Force-Modus berücksichtigen
- `FixedUpdate()` für Physikberechnungen nutzen

Klotz auf schiefer Ebene Ein Klotz der Masse $m = 2kg$ gleitet eine reibungsfreie schiefe Ebene mit Winkel $\theta = 30ř$ hinunter. Berechne die Beschleunigung und implementiere in Unity.

Freikörperdiagramm: Gewichtskraft mg nach unten, Normalkraft N senkrecht zur Oberfläche.
Komponentenanalyse:
- Entlang der Ebene: $mg\sin\theta = ma$
- Senkrecht dazu: $N - mg\cos\theta = 0$

Lösung: $a = g\sin\theta = 9.81 \times \sin(30ř) = 4.905\, m/s^2$
Unity Implementation: Kraft entlang Ebene: $F = mg\sin\theta$, Unity: AddForce mit Richtungsvektor

Kräfte

Grundlegende Wechselwirkungen

Vier fundamentale Kräfte 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 Gravitation und elektromagnetische Kräfte relevant.

Kraft Eine Kraft ist ein Einfluss, der den Bewegungszustand eines Körpers ändert.

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

Einheit: Newton (N) $= 1\,\frac{\text{kg}\cdot\text{m}}{\text{s}^2}$

Gravitationskraft

Newton'sches Gravitationsgesetz
Anziehungskraft zwischen zwei Massen m_1 und m_2 im Abstand R:

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

Gravitationskonstante: $G = 6.67 \cdot 10^{-11}\,\frac{\text{m}^3}{\text{kg}\cdot\text{s}^2}$

Erdbeschleunigung Gewichtskraft eines Körpers der Masse m auf der Erdoberfläche:

$$F_G = m \cdot g$$

Erdbeschleunigung: $g = G \cdot \frac{M_{\text{Erde}}}{R_{\text{Erde}}^2} \approx 9.81\,\frac{\text{m}}{\text{s}^2}$

Mit zunehmender Höhe h:

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

Federkraft

Hooke'sches Gesetz Für eine lineare Feder:

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

- k: Federkonstante (N/m)
- $\vec{x}$: Auslenkung aus der Ruhelage
- Negatives Vorzeichen: Kraft wirkt der Auslenkung entgegen

Spannenergie einer Feder
Potentielle Energie einer gespannten Feder:

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

Harmonischer Oszillator System mit rücktreibender Kraft proportional zur Auslenkung.
Bewegungsgleichung:

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

Lösung:

$$x(t) = x_0 \cdot \cos(\omega t) \quad \text{mit } \omega = \sqrt{\frac{k}{m}}$$

Schwingungsdauer: $T = \frac{2\pi}{\omega}$

Reibungskräfte

Arten der Reibung
Äußere Reibung (Kontaktflächen von Festkörpern): Haftreibung, Gleitreibung, Rollreibung, Wälzreibung, Bohrreibung, Seilreibung
Innere Reibung: zwischen benachbarten Teilchen bei Verformungen

Trockene Reibung (Coulomb-Reibung)

$$\vec{F}_R = \mu \cdot \vec{F}_N$$

- μ: Reibungskoeffizient (dimensionslos)
- $\vec{F}_N$: Normalkraft
- Richtung: entgegengesetzt zur Bewegungsrichtung

Unterscheidung:
- Haftreibung: $\vec{F}_{\text{Haft}} \leq \mu_{\text{Haft}} \cdot \vec{F}_N$
- Gleitreibung: $\vec{F}_{\text{Gleit}} = \mu_{\text{Gleit}} \cdot \vec{F}_N$
- Regel: $\mu_{\text{Haft}} > \mu_{\text{Gleit}}$

Viskose Reibung **Laminare Strömung (Stokes'sche Reibung für Kugel):**

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

(η: Viskosität des Mediums)

Turbulente Strömung:

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

(ρ: Dichte, A: Stirnfläche, c_w: Widerstandsbeiwert)

Unity Implementation

Unity Reibung **Implementierung:**
- Normalkraft: $F_N = mg$ (horizontal)
- Reibung: $F_R = \mu F_N$ entgegen Bewegung
- Geschwindigkeitscheck für Stillstand
- AddForce() in entgegengesetzte Richtung

Unity Luftwiderstand **Turbulente Strömung:**
- $F_R = \frac{1}{2}\rho A c_w v^2$ entgegen Bewegung
- Quadratische Abhängigkeit von Geschwindigkeit
- velocity.normalized für Richtung
- Material-spezifische Parameter

Trägheitskräfte

Beschleunigte Bezugssysteme In beschleunigten Bezugssystemen treten Trägheitskräfte (Scheinkräfte) auf:
Translatorische Trägheitskraft: $\vec{F}_{\text{Trägheit}} = -m \cdot \vec{a}_{\text{System}}$
Zentrifugalkraft bei Rotation: $\vec{F}_{\text{Zentrifugal}} = m \cdot \omega^2 \cdot r \cdot \vec{e}_r$
Corioliskraft bei Bewegung in rotierendem System: $\vec{F}_{\text{Coriolis}} = 2 \cdot m \cdot \vec{v} \times \vec{\omega}$

Trägheitskräfte im Alltag
- Im bremsenden Zug: nach vorne gedrückt fühlen
- In der Kurve: nach außen gedrückt (Zentrifugalkraft)
- Corioliskraft: Ablenkung von Winden und Meeresströmungen
  - Nordhalbkugel: Ablenkung nach rechts
  - Südhalbkugel: Ablenkung nach links

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

Impuls und Stoßgesetze

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

$$\vec{p} = m \cdot \vec{v}$$

Einheit: kg · m/s (vektorielle Größe)

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

$$\vec{I} = \int_{t_1}^{t_2} \vec{F}(t)\, dt = \Delta\vec{p}$$

Für konstante Kraft:

$$\vec{I} = \vec{F} \cdot \Delta t = \vec{p}_{\text{nachher}} - \vec{p}_{\text{vorher}}$$

Impulserhaltung In einem abgeschlossenen System ohne äußere Kräfte bleibt der Gesamtimpuls konstant:

$$\sum \vec{p}_{\text{vorher}} = \sum \vec{p}_{\text{nachher}}$$

Für zwei Körper: $m_1\vec{v}_{1,\text{vorher}} + m_2\vec{v}_{2,\text{vorher}} = m_1\vec{v}_{1,\text{nachher}} + m_2\vec{v}_{2,\text{nachher}}$
Wichtig: Impulserhaltung gilt auch bei dissipativen Vorgängen (z.B. inelastische Stöße).

Elastischer Stoß Sowohl Gesamtimpuls als auch Gesamtenergie bleiben erhalten:
Impulserhaltung:

$$m_1 v_{1,\text{vorher}} + m_2 v_{2,\text{vorher}} = m_1 v_{1,\text{nachher}} + m_2 v_{2,\text{nachher}}$$

Energieerhaltung:

$$\frac{1}{2}m_1 v_{1,\text{vorher}}^2 + \frac{1}{2}m_2 v_{2,\text{vorher}}^2 = \frac{1}{2}m_1 v_{1,\text{nachher}}^2 + \frac{1}{2}m_2 v_{2,\text{nachher}}^2$$

Inelastischer Stoß
Nur der Gesamtimpuls bleibt erhalten, mechanische Energie wird teilweise in andere Formen umgewandelt.
Vollständig inelastischer Stoß: Körper "kleben"nach dem Stoß zusammen.

$$v_{\text{nachher}} = \frac{m_1 v_{1,\text{vorher}} + m_2 v_{2,\text{vorher}}}{m_1 + m_2}$$

Berechnung von Stößen in 1D

Elastischer Stoß

Methode 1: Schwerpunktgeschwindigkeit
1. Schwerpunktgeschwindigkeit:

$$v_{SP} = \frac{m_1 v_1 + m_2 v_2}{m_1 + m_2}$$

2. Geschwindigkeiten nach dem Stoß (Spiegelung):

$$u_1 = 2v_{SP} - v_1$$
$$u_2 = 2v_{SP} - v_2$$

Methode 2: Direkte Formeln

$$u_1 = \frac{(m_1 - m_2)v_1 + 2m_2 v_2}{m_1 + m_2}$$
$$u_2 = \frac{2m_1 v_1 + (m_2 - m_1)v_2}{m_1 + m_2}$$

Vollständig inelastischer Stoß

$$u_1 = u_2 = \frac{m_1 v_1 + m_2 v_2}{m_1 + m_2}$$

Elastischer Stoß mit ungleichen Massen
Leichte Kugel trifft schwere ruhende Kugel ($m_1 \ll m_2$, $v_2 = 0$):

$u_1 \approx -v_1$ (Richtungsumkehr)
$u_2 \approx 0$ (schwerer Körper bleibt fast in Ruhe)

Schwere Kugel trifft leichte ruhende Kugel ($m_1 \gg m_2$, $v_2 = 0$):

$u_1 \approx v_1$ (fast unverändert)
$u_2 \approx 2v_1$ (doppelte Geschwindigkeit)

Impuls- vs. Energieerhaltung
- **Impulserhaltung**: Gilt immer ohne äußere Kräfte (Grundprinzip)
- **Energieerhaltung**:
  Nur für konservative Vorgänge
- **Elastische Stöße**: Beide Erhaltungssätze
- **Inelastische Stöße**: Nur Impulserhaltung
- **Impuls**: Vektorgleichungen (Richtungen wichtig)
- **Energie**: Skalargleichung (nur Beträge)

Anwendungen

Ballistisches Pendel Gerät zur Messung von Projektilgeschwindigkeiten durch vollständig inelastischen Stoß.
Gegeben: Projektilmasse m_K, Pendelmasse m_P, Auslenkung x, Pendellänge L
Projektilgeschwindigkeit:

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

Zweistufiger Prozess:
1. Inelastischer Stoß: Impulserhaltung
2. Pendelschwingung: Energieerhaltung

Raketenantrieb Basiert auf Rückstoßprinzip und Impulserhaltung.
Ziolkowski-Gleichung (maximale Geschwindigkeit):

$$v_{\text{max}} = v_{\text{rel}} \cdot \ln \frac{m_0}{m_{\text{leer}}}$$

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

Unity Implementation

Unity Stoß-Simulation
Elastischer Stoß:
- OnCollisionEnter() für Kollisionserkennung
- Stoßnormale aus Positionsdifferenz
- Geschwindigkeiten in normal/tangential zerlegen
- Elastische Stoßformeln anwenden
- Neue Geschwindigkeiten setzen

Wichtige Erkenntnisse
- Impulserhaltung ist universell gültig
- Kollisionsberechnung erfolgt komponentenweise
- Tangentiale Geschwindigkeiten bleiben bei Stößen unverändert
- Energieverluste nur bei inelastischen Stößen

Arbeit und Energie

Arbeit

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

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

Einheit: Joule (J) $= 1\,\mathrm{N} \cdot \mathrm{m} = 1\,\frac{\mathrm{kg} \cdot \mathrm{m}^2}{\mathrm{s}^2}$
Für konstante Kraft: $W = F \cdot \Delta x \cdot \cos(\alpha)$

Graphische Darstellung Im Kraft-Weg-Diagramm entspricht die Arbeit der Fläche unter der Kurve:

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

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

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

Energie

Energie Energie ist die Fähigkeit eines Systems, Arbeit zu verrichten.

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

Unterschied zu Arbeit:
- Arbeit: Prozessgröße (beschreibt einen Vorgang)
- Energie: Zustandsgröße (charakterisiert einen Zustand)

Energieformen in der Mechanik
Kinetische Energie: $E_{kin} = \frac{1}{2} m v^2$
Potentielle Energie im Schwerefeld: $E_{pot} = mgh$
Spannenergie einer Feder: $E_{spann} = \frac{1}{2} k x^2$
Rotationsenergie: $E_{rot} = \frac{1}{2} J \omega^2$

Energieerhaltung In einem abgeschlossenen System ohne nicht-konservative Kräfte bleibt die Gesamtenergie konstant:

$$E_{ges} = E_{kin} + E_{pot} + E_{spann} + E_{rot} = \text{const.}$$

Konservative vs. nicht-konservative Kräfte:
- **Konservativ**: Arbeit ist wegunabhängig (Gravitation, Federkraft)
- **Nicht-konservativ**: Arbeit hängt vom Weg ab (Reibung, Luftwiderstand)

Energiewandlungen beim Pendel
Pendel der Masse m und Länge l zeigt Umwandlung zwischen potentieller und kinetischer Energie:
Am höchsten Punkt: $E_{pot,max} = mgl(1 - \cos\alpha)$
Am tiefsten Punkt: $E_{kin,max} = \frac{1}{2} m v_{max}^2 = mgl(1 - \cos\alpha)$
Gesamtenergie bleibt konstant: $E_{ges} = E_{pot} + E_{kin}$

Leistung und Wirkungsgrad

Leistung
P ist die pro Zeiteinheit verrichtete Arbeit:

$$P = \frac{dW}{dt} = \vec{F} \cdot \vec{v}$$

Einheit: Watt (W) $= 1\,\frac{\mathrm{J}}{\mathrm{s}} = 1\,\frac{\mathrm{kg} \cdot \mathrm{m}^2}{\mathrm{s}^3}$
Für Rotationsbewegung: $P = \tau \omega$

Wirkungsgrad
Verhältnis Nutzenergie zu zugeführter Energie:

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

Dimensionslose Zahl zwischen 0 und 1 (oder 0-100%).

Impuls und Drehimpuls

Linearer Impuls $\vec{p} = m\vec{v}$
Newton'sches Gesetz: $\vec{F} = \frac{d\vec{p}}{dt}$

Kraftstoß Änderung des Impulses durch Kraft über Zeit:

$$\vec{J} = \int_{t_1}^{t_2} \vec{F}\,dt = \Delta\vec{p} = m\vec{v}_f - m\vec{v}_i$$

Für konstante Kraft: $\vec{J} = \vec{F}\Delta t$

Impulserhaltung Ohne äußere Kräfte bleibt der Gesamtimpuls konstant:

$$\sum \vec{p}_i = \sum \vec{p}_f$$

Für Zwei-Körper-System:

$$m_1\vec{v}_{1i} + m_2\vec{v}_{2i} = m_1\vec{v}_{1f} + m_2\vec{v}_{2f}$$

Drehimpuls **Für Punktteilchen:**

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

Für starren Körper:

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

Drehimpulserhaltung Ohne äußere Drehmomente:

$$\frac{d\vec{L}}{dt} = \vec{\tau}_{ext} = 0 \Rightarrow \vec{L} = \text{const.}$$

Anwendungen: Eiskunstläufer, Planetenbewegung, Kreisel

Kollisionen

Kollisionstypen
Elastisch: Impuls und kinetische Energie erhalten
Inelastisch: Nur Impuls erhalten
Vollständig inelastisch: Objekte kleben nach Kollision zusammen

Elastische Kollision (1D)
Für zwei Objekte mit Massen m_1, m_2 und Anfangsgeschwindigkeiten v_{1i}, v_{2i}:

$$v_{1f} = \frac{(m_1 - m_2)v_{1i} + 2m_2 v_{2i}}{m_1 + m_2}$$
$$v_{2f} = \frac{(m_2 - m_1)v_{2i} + 2m_1 v_{1i}}{m_1 + m_2}$$

Unity Implementation

Unity Energieberechnung **Energieformen:**
- Kinetisch: $\frac{1}{2} m v^2$ mit rb.velocity.sqrMagnitude
- Potentiell (Gravitation): mgh mit position.y
- Elastisch: $\frac{1}{2} k x^2$ mit displacement
- Monitoring in Update() für Debugging

Unity Kollision (1D) **Elastische Kollision:**
- Formeln: $v_{1f} = \frac{(m_1 - m_2)v_{1i} + 2m_2 v_{2i}}{m_1 + m_2}$
- OnCollisionEnter() Event nutzen
- Massen und Geschwindigkeiten auslesen
- Neue Geschwindigkeiten direkt setzen

Unity Drehimpuls **Berechnung:**
- $\vec{L} = \vec{r} \times \vec{p}$
- Vector3.Cross() für Kreuzprodukt
- Position relativ zu Pivot-Punkt
- Linearer Impuls $\vec{p} = m\vec{v}$

Energie-Impuls-Problemlösung

Schritt 1: System und Randbedingungen identifizieren
- Systemgrenzen definieren
- Konservative und nicht-konservative Kräfte identifizieren
- Erhaltungsgesetze prüfen

Schritt 2: Passenden Erhaltungssatz wählen
- Energieerhaltung für Höhen-, Federprobleme
- Impulserhaltung für Kollisionsprobleme
- Beide für komplexe mehrstufige Probleme

Schritt 3: Gleichungen aufstellen
- Anfangs- und Endzustände beschreiben
- Erhaltungsprinzipien anwenden
- Zusätzliche Randbedingungen einbeziehen

Schritt 4: Lösen und verifizieren
- Algebraisch lösen vor Zahleneinsetzung
- Einheiten und physikalische Plausibilität prüfen
- Erhaltungsgesetze verifizieren

Kollisionsanalyse Zwei Autos: Auto A (1000 kg) mit 20 m/s, Auto B (1500 kg) mit -15 m/s. Nach Kollision bewegt sich Auto A mit 5 m/s. Berechne Auto B's Endgeschwindigkeit und Energieverlust.

Gegeben: $m_A = 1000kg$, $m_B = 1500kg$, $v_{Ai} = 20m/s$, $v_{Bi} = -15m/s$, $v_{Af} = 5m/s$
Impulserhaltung: $1000(20) + 1500(-15) = 1000(5) + 1500v_{Bf}$
$20000 - 22500 = 5000 + 1500v_{Bf}$
$v_{Bf} = -5m/s$
Energieanalyse: $KE_i = \frac{1}{2}(1000)(20^2) + \frac{1}{2}(1500)(15^2) = 368750J$
$KE_f = \frac{1}{2}(1000)(5^2) + \frac{1}{2}(1500)(5^2) = 31250J$
$\Delta E = 368750 - 31250 = 337500J$ verloren

Projektilbewegung mit Energie Ein Projektil wird mit Winkel $\theta = 45$ř und Anfangsgeschwindigkeit $v_0 = 20m/s$ abgeschossen. Maximale Höhe mit Energieerhaltung berechnen.

Energiemethode: Start: $E_i = \frac{1}{2}mv_0^2$ (Boden als Referenz)
Max. Höhe: $E_f = mgh_{max} + \frac{1}{2}mv_x^2$
Da $v_x = v_0 \cos\theta$ konstant bleibt:
$\frac{1}{2}mv_0^2 = mgh_{max} + \frac{1}{2}m(v_0 \cos\theta)^2$
$h_{max} = \frac{v_0^2 \sin^2\theta}{2g} = \frac{(20)^2 \sin^2(45ř)}{2(9.81)} = 10.2m$

Rotation

Kinematik der Rotation

Rotationsbewegung Eine Rotationsbewegung ist die Drehung eines Körpers um eine bestimmte Achse. Die Beschreibung erfolgt analog zur Translation mit rotatorischen Größen:

- 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.

Rotatorische Kinematik

Analog zur Translation gelten folgende Beziehungen:

$$\omega = \frac{d\varphi}{dt}$$
$$\alpha = \frac{d\omega}{dt} = \frac{d^2\varphi}{dt^2}$$
$$\varphi = \int \omega\, dt$$
$$\omega = \int \alpha\, dt$$

Für konstante Winkelbeschleunigung:

$$\omega(t) = \omega_0 + \alpha t$$
$$\varphi(t) = \varphi_0 + \omega_0 t + \frac{1}{2}\alpha t^2$$
$$\omega^2 = \omega_0^2 + 2\alpha(\varphi - \varphi_0)$$

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:

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

Umrechnung zwischen Grad und Radian: $\varphi_{rad} = \varphi_{deg} \cdot \frac{\pi}{180}$
Wichtige Werte:

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

Zusammenhang zwischen linearer und Rotationsbewegung
Bei einer Kreisbewegung mit Radius r gelten folgende Beziehungen:

$$s = r\varphi \quad \text{(Bogenlänge)}$$
$$v = r\omega \quad \text{(Bahngeschwindigkeit)}$$
$$a_t = r\alpha \quad \text{(Tangentialbeschleunigung)}$$
$$a_c = \frac{v^2}{r} = r\omega^2 \quad \text{(Zentripetalbeschleunigung)}$$

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

Schwerpunkt und Trägheitsmoment

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

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

Für kontinuierliche Massenverteilungen:

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

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 I ist ein Maß für den Widerstand eines Körpers gegenüber Rotationsbeschleunigungen:

$$I = \sum_{i=1}^{n} m_i \cdot r_i^2$$

Für kontinuierliche Massenverteilungen:

$$I = \iiint_K r^2\,\rho(\vec{r})\, dV = \int r^2\, dm$$

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

Trägheitsmomente wichtiger Körper
Punktmasse: $I = mr^2$
Dünner Stab (Länge L, Achse durch Mitte):
$I = \frac{1}{12}mL^2$
Dünner Stab (Achse am Ende): $I = \frac{1}{3}mL^2$
Vollzylinder (Radius R, Symmetrieachse):
$I = \frac{1}{2}mR^2$
Hohlzylinder (Symmetrieachse): $I = mR^2$
Vollkugel (Radius R, Achse durch Mittelpunkt):
$I = \frac{2}{5}mR^2$

Steiner'scher Satz (Parallelachsentheorem) Für eine Achse parallel zur Schwerpunktachse im Abstand d:

$$I = I_S + md^2$$

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

Dynamik der Rotation

Drehmoment Das Drehmoment $\vec{\tau}$ ist die Ursache einer Rotationsbeschleunigung:

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

Der Betrag des Drehmoments ist:

$$\tau = rF\sin\phi = Fd$$

wobei ϕ der Winkel zwischen Hebelarm und Kraft ist und d der senkrechte Abstand der Kraftwirkungslinie von der Drehachse.

Newton'sches Gesetz für Rotationen Analog zum zweiten Newton'schen Gesetz für Translation:

$$\sum \vec{\tau} = I\vec{\alpha}$$

Diese Gleichung verknüpft das resultierende Drehmoment mit der Winkelbeschleunigung.

Drehimpuls Der Drehimpuls $\vec{L}$ ist das rotatorische Analogon zum linearen Impuls:

$$\vec{L} = I\vec{\omega}$$

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

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

Das Drehmoment ändert den Drehimpuls gemäß:

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

Drehimpulserhaltung In Abwesenheit äußerer Drehmomente bleibt der Drehimpuls konstant:

$$\frac{d\vec{L}}{dt} = \vec{\tau}_{ext} = 0 \Rightarrow \vec{L} = \text{const.}$$

Anwendungen:
- Eiskunstläufer, der die Arme anzieht, um schneller zu rotieren
- Stabilität von Fahrrädern aufgrund rotierenden Räder
- Präzession eines Kreisels
- Planetenbewegung

Rotationsenergie
Die kinetische Energie der Rotation:

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

Für kombinierte Translation und Rotation:

$$E_{ges} = \frac{1}{2}mv_{cm}^2 + \frac{1}{2}I_{cm}\omega^2$$

Rollbedingung ohne Schlupf: $v_{cm} = R\omega$
Dann: $E_{ges} = \frac{1}{2}mv^2(1 + \frac{I}{mR^2})$

Arbeit und Leistung bei Rotation
Rotationsarbeit:

$$W = \int \tau \, d\varphi$$

Für konstantes Drehmoment: $W = \tau\Delta\varphi$
Rotationsleistung: $P = \tau\omega$
Arbeitssatz für Rotation:

$$W_{net} = \Delta E_{rot} = \frac{1}{2}I\omega_f^2 - \frac{1}{2}I\omega_i^2$$

Orientierung im Raum und Quaternionen

Euler-Winkel Euler-Winkel beschreiben eine Rotation im dreidimensionalen Raum durch drei Winkel:
- Pitch (φ): Drehung um die x-Achse
- Yaw (θ): Drehung um die y-Achse
- Roll (ψ): Drehung um die z-Achse

Wichtig: Die Reihenfolge der Drehungen ist entscheidend!
- Intrinsische Drehungen: Achsen drehen sich mit dem Objekt
- Extrinsische Drehungen: Achsen bleiben fixiert

Quaternionen in Unity Unity verwendet Quaternionen zur Darstellung von Rotationen wegen ihrer Vorteile:
- Keine Gimbal-Lock-Probleme
- Effizientere Interpolation zwischen Rotationen
- Numerisch stabiler

Quaternionen haben die Form $q = w + xi + yj + zk$, wobei $i^2 = j^2 = k^2 = ijk = -1$.
Rotation um Achse $\vec{n}$ mit Winkel α:

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

Standfestigkeit und Gleichgewicht

Standfestigkeit
Ein Körper ist standfest, wenn die Wirkungslinie der Gewichtskraft durch die Unterstützungsfläche verläuft.
Gleichgewichtsarten:
- **Stabil**: Rückkehr zur Ausgangslage (Energie-Minimum)
- **Labil**: Entfernung von Ausgangslage (Energie-Maximum)
- **Indifferent**: Verharren in neuer Position (konstante Energie)

Kippkriterium
Ein Körper kippt, wenn die Wirkungslinie der Gewichtskraft außerhalb der Unterstützungsfläche verläuft.
Für einen Quader (Breite b, Höhe h) auf geneigter Ebene (Winkel α):

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

Unity Implementation

Unity Rotation **Implementierung:**
- $\omega = \omega_0 + \alpha t$ für Winkelgeschwindigkeit
- transform.Rotate() für kinematische Rotation
- Rigidbody.angularVelocity für Physik-Rotation
- Deg2Rad für Einheitenumrechnung

Unity Drehmoment-Berechnung **Implementierung:**
- $\vec{\tau} = \vec{r} \times \vec{F}$ mit Vector3.Cross()
- Hebelarm von Zentrum zu Kraftangriffspunkt
- AddTorque() für Anwendung
- Debug.DrawRay() für Visualisierung

Unity Quaternionen **Wichtige Operationen:**
- Quaternion.Euler() aus Winkeln
- Quaternion.AngleAxis() aus Achse+Winkel
- Multiplikation für Kombination
- Quaternion.Slerp() für Interpolation

Unity Rotationsenergie **Berechnung:**
- $E_{rot} = \frac{1}{2}I\omega^2$ mit angularVelocity.magnitude
- Trägheitsmoment für Standardformen approximieren
- Gesamt: $E = E_{trans} + E_{rot}$
- Monitoring für Energieerhaltung

Rotationsprobleme lösen

Schritt 1: Rotationsachse identifizieren
- Feste Achse (Türscharnier) oder bewegliche Achse (rollender Ball)
- Koordinatensystem mit Achse senkrecht zur Bewegungsebene wählen

Schritt 2: Trägheitsmoment berechnen
- Standardformeln für einfache Formen verwenden
- Steiner'schen Satz anwenden wenn nötig
- Für zusammengesetzte Objekte einzelne Momente summieren

Schritt 3: Kräfte und Drehmomente analysieren
- Drehmoment berechnen: $\tau = rF\sin\phi$
- $\sum \tau = I\alpha$ für Dynamik anwenden
- Randbedingungen berücksichtigen (Rollen, feste Achse, etc.)

Schritt 4: Erhaltungsgesetze anwenden
- Energieerhaltung für Objekte auf schiefen Ebenen
- Drehimpulserhaltung für isolierte Systeme
- Kombinierte linear-rotatorische Analyse für komplexe Bewegungen

Kugel rollt schiefe Ebene hinunter Eine Vollkugel (Radius R, Masse m) rollt ohne Schlupf eine schiefe Ebene mit Winkel θ hinunter. Berechne die Beschleunigung des Schwerpunkts.
Gegeben: $I = \frac{2}{5}mR^2$ für Vollkugel, Rollbedingung $a = R\alpha$
Kräfteanalyse:
- Entlang der Ebene: $mg\sin\theta - f = ma$
- Drehmoment um Schwerpunkt: $fR = I\alpha = \frac{2}{5}mR^2 \cdot \frac{a}{R}$

Aus Drehmomentgleichung: $f = \frac{2}{5}ma$
Einsetzen in Kraftgleichung: $mg\sin\theta - \frac{2}{5}ma = ma$
$mg\sin\theta = ma(1 + \frac{2}{5}) = \frac{7}{5}ma \to a = \frac{5g\sin\theta}{7} = 0.714 \times g\sin\theta$
Ergebnis: Die Beschleunigung ist kleiner als beim Rutschen ($g\sin\theta$), da Energie in Rotation fließt.

Tür-Simulation **Konzept:** Realistische Türsimulation mit Drehmoment

Wichtige Komponenten:
- Drehmoment für Öffnung: AddTorque(Vector3.up * maxTorque)
- Dämpfung: -dampingCoefficient * angularVelocity
- Winkelbegrenzung mit eulerAngles und maxAngle
- Input-gesteuerte Aktivierung

Maximaler Überhang gestapelter Objekte Ein Turm aus n identischen Bausteinen der Länge L kann maximal überhängen um:

$$\text{Überhang}_{\text{max}} = \frac{L}{2} \cdot \sum_{i=1}^{n} \frac{1}{i} = \frac{L}{2}(1 + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n})$$

Dies basiert auf der harmonischen Reihe und zeigt, dass theoretisch unbegrenzte Überhänge möglich sind (divergente Reihe).

Erweiterte Themen und Unity-Implementierung

Unity Physics System

Unity Physics Pipeline

Unity's Physiksimulation läuft in diskreten Schritten:
- `FixedUpdate()`: Wird in festen Intervallen aufgerufen (Standard 50Hz)
- Physikberechnungen zwischen FixedUpdate-Aufrufen
- `Update()`: Wird einmal pro Frame aufgerufen (variable Rate)
- **Regel**: FixedUpdate für physikbezogenen Code verwenden

Zeitsteuerung:
- `Time.fixedDeltaTime`: Zeitschritt für Physik (Standard: 0.02s = 50Hz)
- `Time.deltaTime`: Zeit seit letztem Frame (variabel)
- `Time.timeScale`: Globaler Zeitfaktor (für Slow-Motion etc.)

Rigidbody-Komponenten-Eigenschaften

Grundlegende Eigenschaften:
- **Mass**: Objektmasse in kg
- **Drag**: Linearer Dämpfungskoeffizient (Luftwiderstand)
- **Angular Drag**: Rotationsdämpfungskoeffizient
- **Use Gravity**: Reagiert auf globale Gravitation
- **Is Kinematic**: Physik-gesteuert vs. Skript-gesteuert

Erweiterte Eigenschaften:
- **Interpolate**: Glättung für visuelle Darstellung
- **Collision Detection**: Continuous, Discrete, etc.
- **Constraints**: Einschränkung Position/Rotation
- **Center of Mass**: Schwerpunkt-Position

Unity Physics Setup

Wichtige Konfigurationen:
- Physics.gravity für globale Gravitation
- Time.fixedDeltaTime für Physik-Zeitschritt
- Rigidbody: mass, drag, angularDrag
- CollisionDetectionMode.Continuous für schnelle Objekte
- RigidbodyInterpolation für glatte Bewegung

Wichtige Erkenntnisse

Theoretische Grundlagen:
- Physikalische Gesetze gelten universell
- Erhaltungsgesetze zentral für Problemlösungen
- Numerische Integration beeinflusst Genauigkeit
- Reibung und Dämpfung sind realitätsnah wichtig

Unity-spezifische Aspekte:
- Fixed Timestep ist entscheidend für Stabilität
- Physics Materials beeinflussen Verhalten stark
- Performance-Optimierung ist für komplexe Szenen notwendig
- Debugging-Tools helfen bei der Problemanalyse

Kugel-Kollisions-Simulation

Problem: Rollende Kugel auf Schräge kollidiert elastisch

Physik-Berechnung:
- Rollbeschleunigung: $a = \frac{5g\sin\theta}{7} = 3.51\,m/s^2$
- Endgeschwindigkeit: $v = \sqrt{2ad} = 3.74\,m/s$
- Elastische Kollision: Geschwindigkeiten tauschen

Unity-Konzept:
- Schräge mit PhysicMaterial (hohe Reibung)
- Zwei Kugeln mit elastischem Material
- OnCollisionEnter() für Geschwindigkeitstausch
- Debug-Ausgabe für Energieerhaltung

Unity Kraft-Modi

Unity Kraftanwendung

Kraftmethoden:
- **AddForce()**: Standard-Kraftanwendung
- **AddForceAtPosition()**: Kraft + Drehmoment
- **AddTorque()**: Nur Rotation
- **AddExplosionForce()**: Radiale Explosion

Force-Modi:
- **Force**: $F = ma$ (mit Masse/Zeit)
- **Acceleration**: a direkt (ohne Masse)
- **Impulse**: $J = F\Delta t$ (mit Masse)
- **VelocityChange**: Δv direkt (ohne Masse)

Erweiterte Kraftsteuerung

Controller-Konzept:
- Input-basierte Kraftanwendung
- Bewegung: transform.forward * thrustForce
- Rotation: AddTorque mit transform.up/right
- Explosion: AddExplosionForce für Umgebung
- Geschwindigkeitsbegrenzung mit velocity.normalized

Unity Kollisionen

Unity Kollisions-Events

Events:
- **Collision**: OnCollisionEnter/Stay/Exit
- **Trigger**: OnTriggerEnter/Stay/Exit

Informationen:
- **contacts**: Kontaktpunkte
- **impulse**: Stoßimpuls
- **relativeVelocity**: Relativgeschwindigkeit
- **normal**: Oberflächennormale

Erweiterte Kollisions-Analyse

Reaktions-System:
- Kollisionsgeschwindigkeit aus relativeVelocity
- Material-spezifische Restitution (Tags)
- Audio/Partikel basierend auf Intensität
- Spezielle Reaktionen: Bouncy, Sticky, Breakable
- Vector3.Reflect() für elastische Reflektion

Custom Physics

Numerische Integrationsmethoden

Unity verwendet implizite Euler-Integration, aber benutzerdefinierte Implementierungen können verwenden:
- **Expliziter Euler**: Einfach, aber potentiell instabil
- **Verlet-Integration**: Bessere Energieerhaltung
- **Runge-Kutta**: Höhere Genauigkeit für komplexe Systeme
- **Leapfrog**: Gut für Orbitalprobleme

Vergleich der Methoden:
- **Stabilität**: Verlet > Runge-Kutta > Euler
- **Genauigkeit**: Runge-Kutta > Verlet > Euler
- **Performance**: Euler > Verlet > Runge-Kutta
- **Energieerhaltung**: Verlet > Runge-Kutta > Euler

Custom Physics Integration

Integrationsmethoden:
- **Euler**: $v+ = a\Delta t$, $x+ = v\Delta t$
- **Verlet**: $x_{new} = 2x - x_{old} + a\Delta t^2$
- **Runge-Kutta**: Höhere Genauigkeit, 4 Evaluationen
- Kraftberechnung: Gravitation + Feder + Dämpfung
- LineRenderer für Trajektorie

Performance-Tipps

Physik-Performance-Tipps

Kollisions-Optimierung:
- Passende CollisionDetectionMode wählen
- Primitive Collider bevorzugen
- Layer-Matrix für Kollisions-Filtering

Rigidbody-Optimierung:
- Kinematic für statische Objekte
- Object Pooling für viele Objekte
- Fixed Timestep anpassen

Object Pooling

Performance-Optimierung:
- Queue<GameObject> für Pool-Verwaltung
- GetObject() / ReturnObject() API
- Automatisches Cleanup nach Zeit/Position
- Rigidbody-Zustand zurücksetzen (Sleep/WakeUp)
- Prefab-basierte Instanziierung

Prüfungstipps

Unity Physics Prüfungsstrategie

Kernphysik-Konzepte beherrschen

- Newton'sche Gesetze und ihre Anwendungen
- Energie- und Impulserhaltung verstehen
- Rotationsdynamik-Berechnungen üben
- Wichtige Formeln und ihre Herleitungen kennen
- Vektormathematik und Koordinatensysteme

Unity-Implementierungs-Wissen

- Rigidbody-Komponenten-Eigenschaften und -Methoden
- Kraftanwendungstechniken und Force-Modi
- Kollisionserkennung und -reaktion
- Koordinatensystem-Transformationen
- Physics Material und Reibungsmodelle

Problemlösungsansatz

- Beteiligte physikalische Prinzipien identifizieren
- Koordinatensystem und Freikörperdiagramme aufstellen
- Erhaltungsgesetze anwenden wo angemessen
- In Unity-Implementierungskonzepte übersetzen
- Code-Snippets für häufige Szenarien memorieren

Häufige Prüfungsthemen

- Projektilbewegung mit Unity-Vektoren
- Kollisionsanalyse und Impulserhaltung
- Rotationsbewegung und Drehmoment-Berechnungen
- Energieerhaltung in mechanischen Systemen
- Custom Physics Implementation
- Performance-Optimierung in Unity Physics