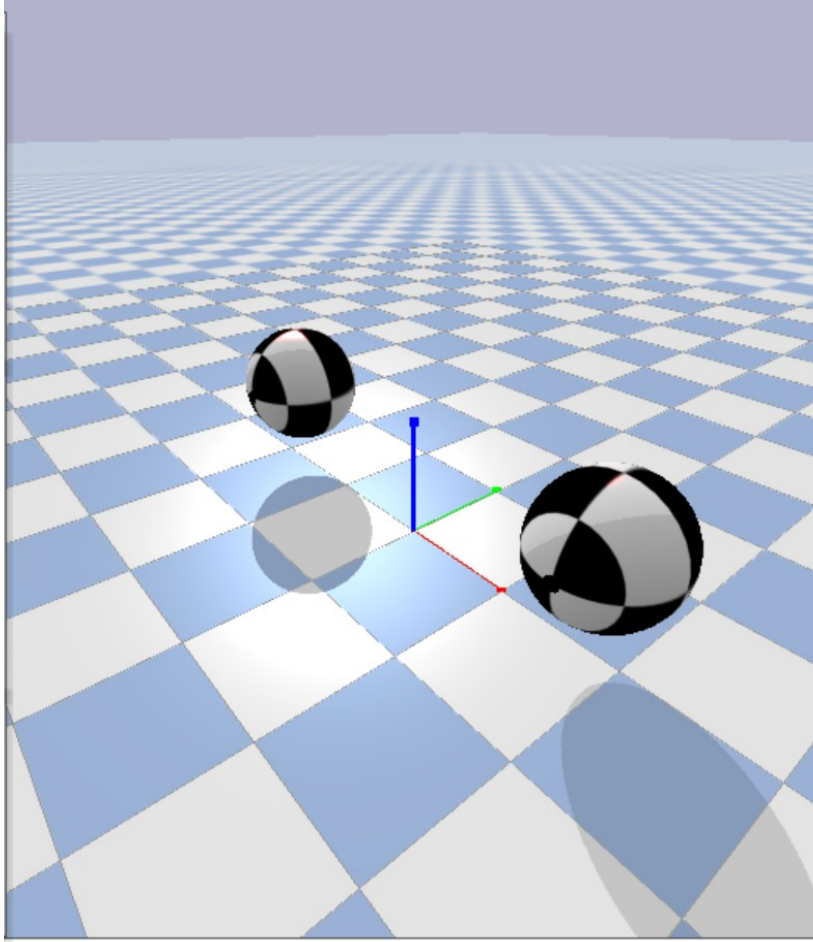
Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

FAKULTÄT
FÜR MATHEMATIK, INFORMATIK
UND NATURWISSENSCHAFTEN



Impulse-based Game Physics

Pascal Spethmann

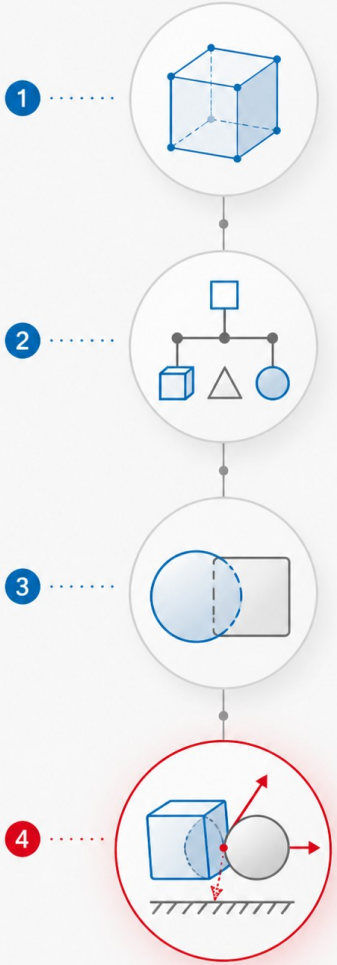
03.06.2026

Agenda

1. Einordnung
2. Rigid Bodies & Simulation Loop
3. Bewegung & Numerical Integration
4. Collision Detection als Grundlage
5. Impulse-based Collision Response
6. Contact Constraints, Restitution, Friction
7. Pybullet-Demo

1

Einordnung

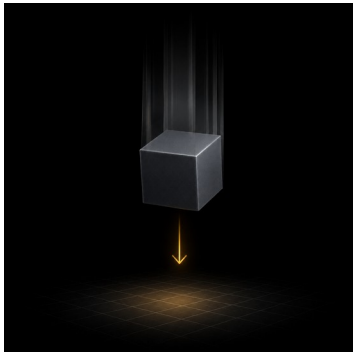


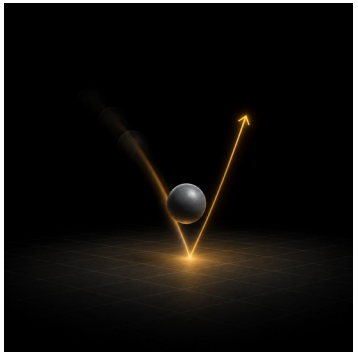
Warum Impulse-based Game Physics?

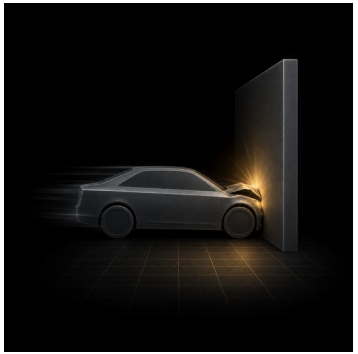
1. Glaubwürdigkeit
   Objekte sollen sich so bewegen, wie wir es erwarten

2. Echtzeit
   Die Simulation muss in jedem Frame schnell genug berechnet werden

3. Kontrolle
   Für Spiele plausibles Verhalten oft wichtiger als perfekte Physik

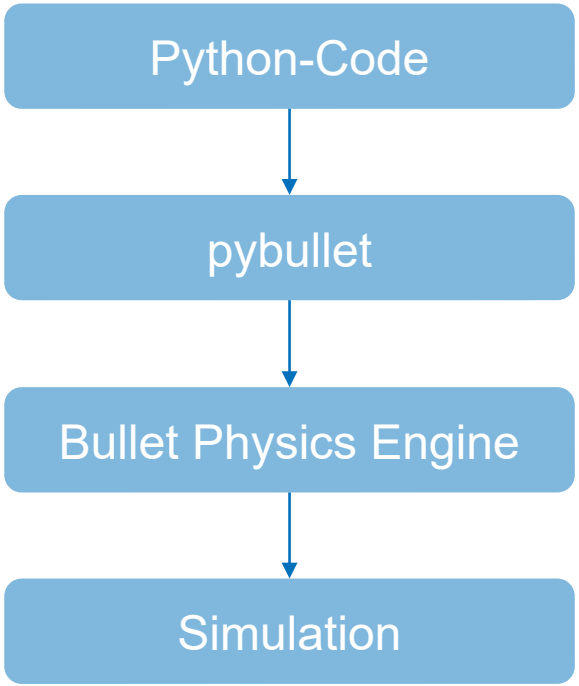






Was ist Bullet?

- Rigid Body Simulation
  Bewegung fester Körper wie Würfel, Kugeln oder Fahrzeuge
- Collision Detection
  Erkennt, wann und wo Objekte miteinander kollidieren
- Constraints
  Verbindungen und Einschränkungen, z.B. Gelenke, Achsen oder Limits

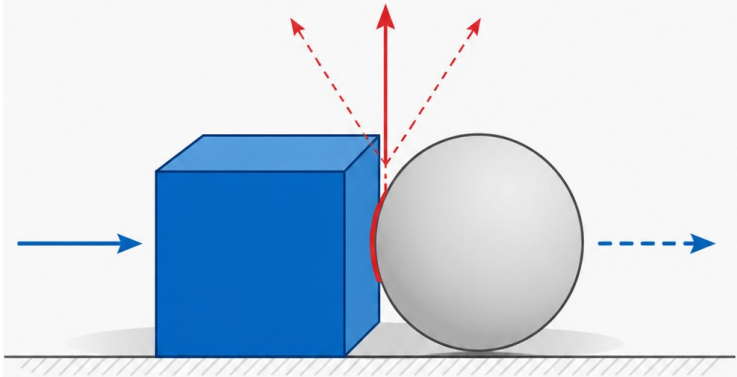


Einordnung in den Seminarplan

Collision Detection erkennt den Kontakt – Game Physics berechnet die Reaktion.

2

Rigid Bodies & Simulation Loop

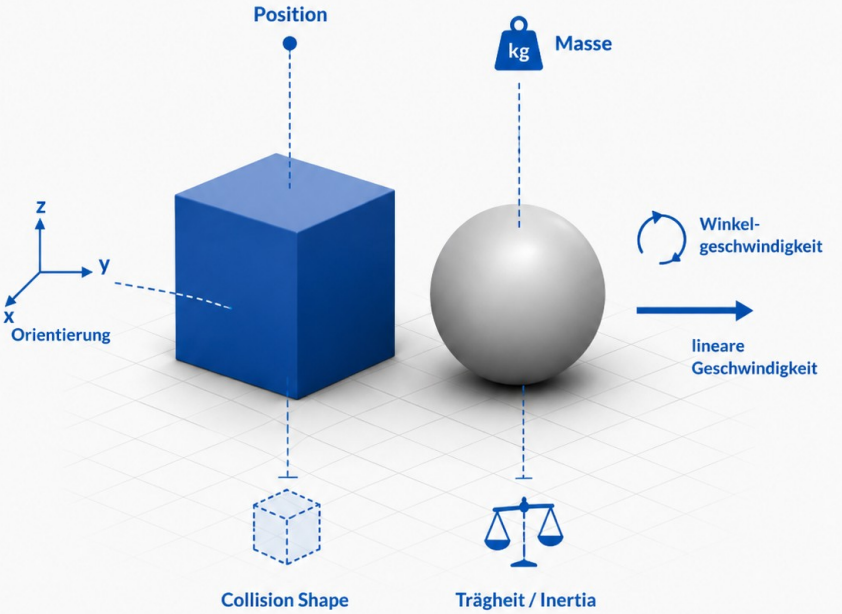


Was ist ein Rigid Body

Ein **Rigid Body** ist ein **Körper**, der sich **nicht verformt.**

Eigenschaften:

- Position
- Orientierung
- Masse
- Lineare Geschwindigkeit
- Winkelgeschwindigkeit
- Trägheit
- Collision Shape

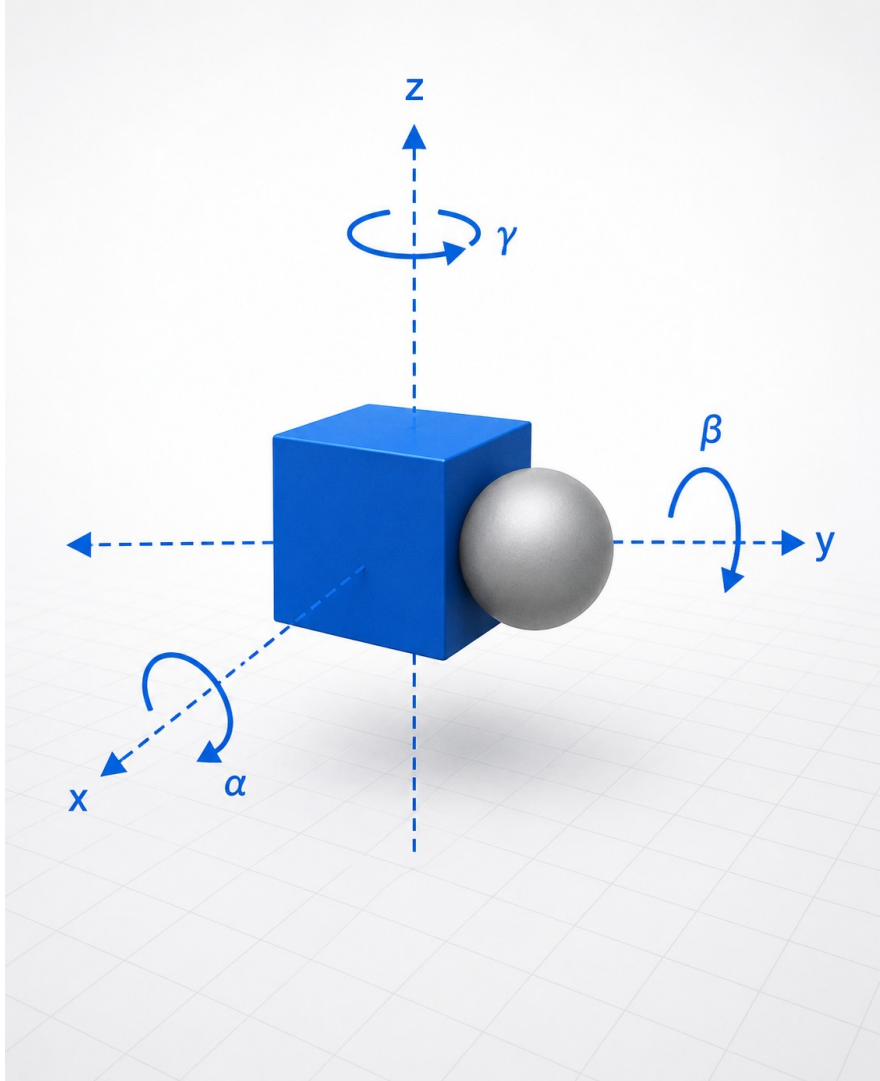


6 Degrees of Freedom

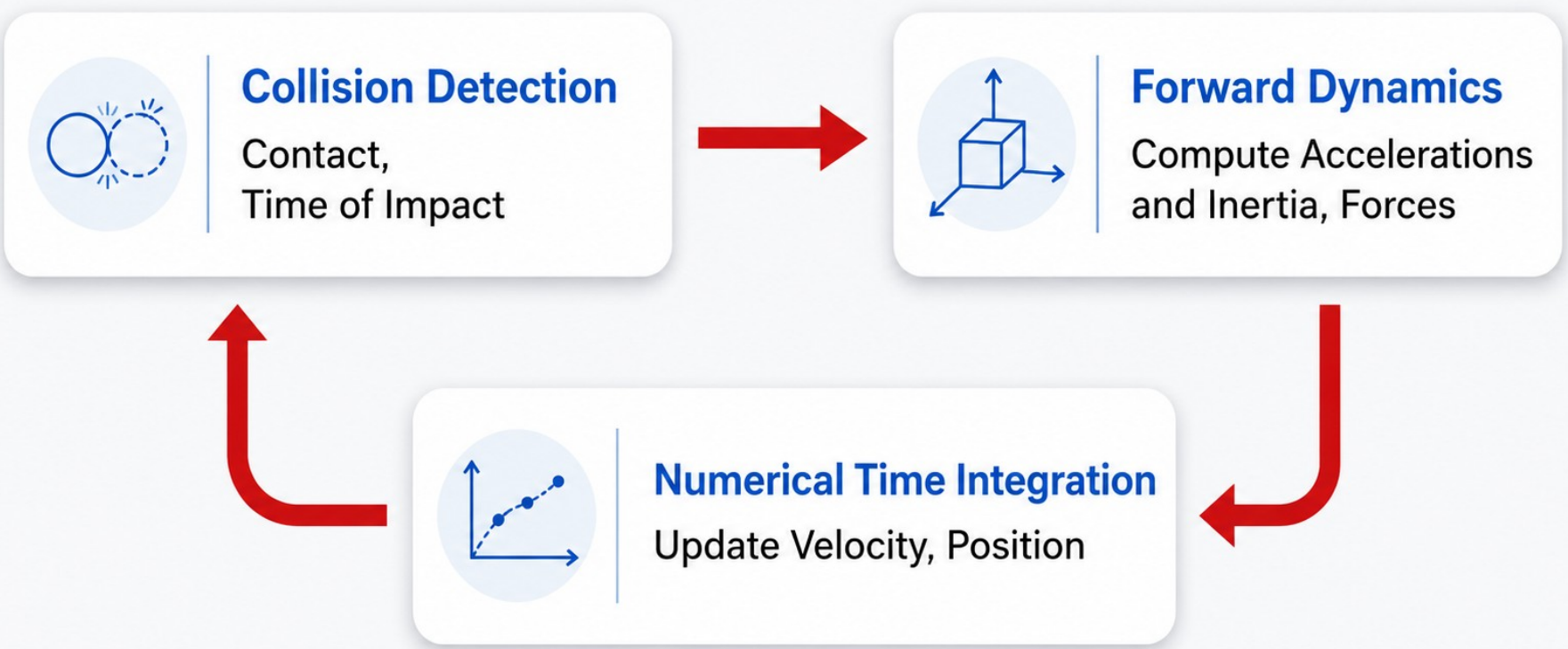
Ein Rigid Body in 3D hat:

- 3 Freiheitsgrade für Translation:
  x, y, z

- 3 Freiheitsgrade für Rotation:
  α, β, γ

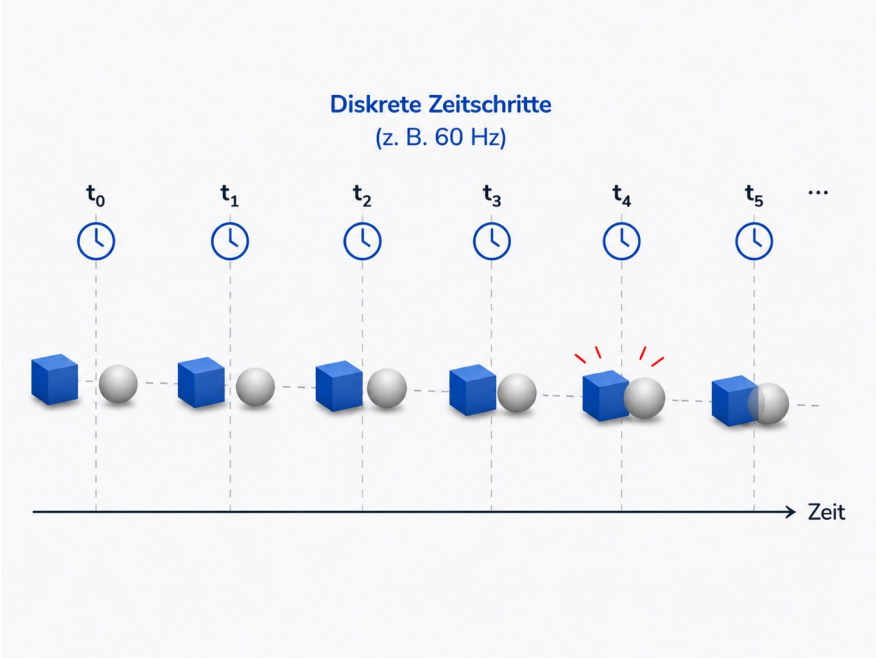


Rigid Body Simulation Loop



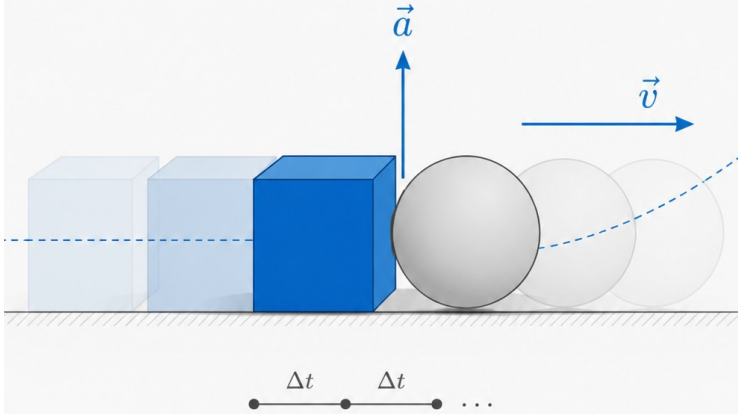
Game Physics ist diskret

- Simulation läuft nicht kontinuierlich
- Typisch sind 60Hz, 120Hz oder 240Hz
- Pro Zeitschritt wird approximiert
- Dadurch entstehen Fehler wie Penetration



3

Bewegung & Numerical Integration



Von Kraft zu Bewegung

Was macht die Engine:

1. Kräfte sammeln
2. Beschleunigung berechnen
3. Geschwindigkeit aktualisieren
4. Position verändern

Grundidee:

- F = m * a
- a = F / m

Von Newton zu Impulsen

- Newtons 2. Gesetz
- Kraft erzeugt Beschleunigung
- Beschleunigung verändert Geschwindigkeit
- Kollisionen werden als sehr kurze Kräfte modelliert

Formeln:

- $F = m * a$
- $J = F * \Delta t$
- $J = m * \Delta v$

Numerical Integration

Vereinfacht:

- velocity += acceleration * dt
- position += velocity * dt

Erklärung:

- dt ist der Zeitschritt
- Geschwindigkeit ist Änderung der Position
- Beschleunigung ist Änderung der Geschwindigkeit

Beschleunigung a -> Geschwindigkeit v -> Position x

Symplectic Euler

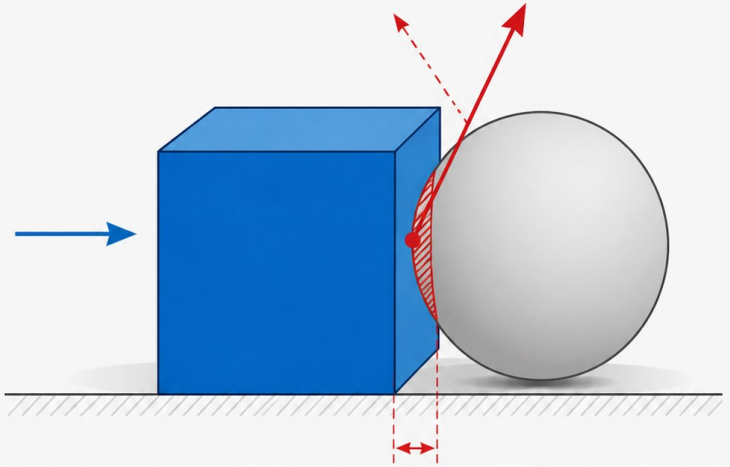
- Einfach
- Schnell
- Stabiler als explicit Euler für viele Fälle
- Gut für Echtzeit

$$v(t + dt) = v(t) + a * dt$$

$$x(t + dt) = x(t) + v(t + dt) * dt$$

4

Collision Detection als Grundlage



Collision Detection vs. Collision Response

Collision Detection

- Berühren sich zwei Körper?
- Wo berühren Sie sich?
- Wie tief dringen sie ein?
- Was ist die Kontakt-Normale?

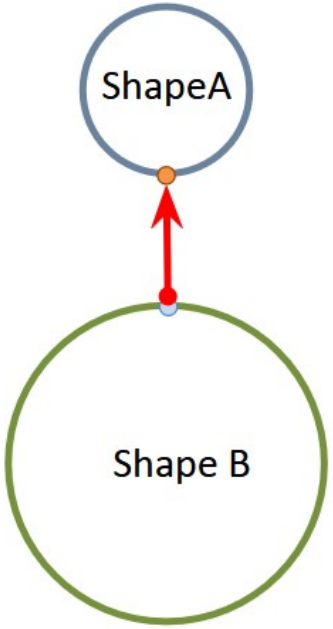
Collision Response

- Welche Geschwindigkeiten müssen geändert werden?
- Welcher Impuls ist nötig?
- Wie stark prallt ein Körper ab?
- Wie wirkt Reibung?

Contact Information

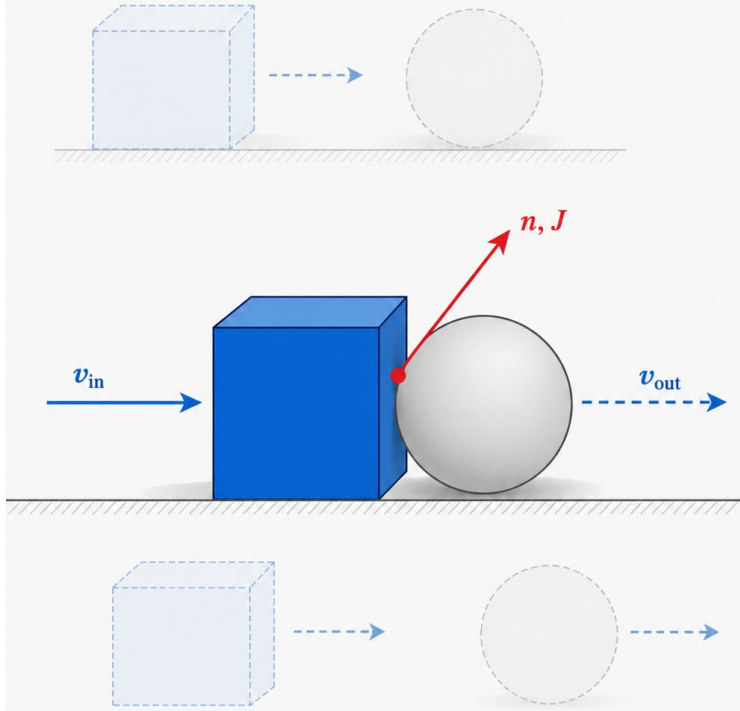
Größe	Bedeutung
point on A	Kontaktpunkt auf Körper A
point on B	Kontaktpunkt auf Körper B
normale	Trennungsrichtung
distance	Abstand oder Eindringtiefe

- distance > 0 : Körper sind getrennt
- distance = 0 : Körper berühren sich
- distance < 0 : Körper dringen ineinander ein



5

Impulse-based Collision Response



Von Kraft zu Bewegung

Grundidee

Ein Impuls ist ein kurzer Stoß, der Geschwindigkeit direkt verändert.

Formelintuition

$$\text{Impulse} = F \cdot \Delta t$$
$$\text{Impulse} = m \cdot \Delta v$$
$$\Delta v = \text{Impulse} / m$$

Gleicher Impuls: leichte Körper ändern ihre Geschwindigkeit stärker als schwere Körper.

Conservation of Momentum (Impulserhaltung)

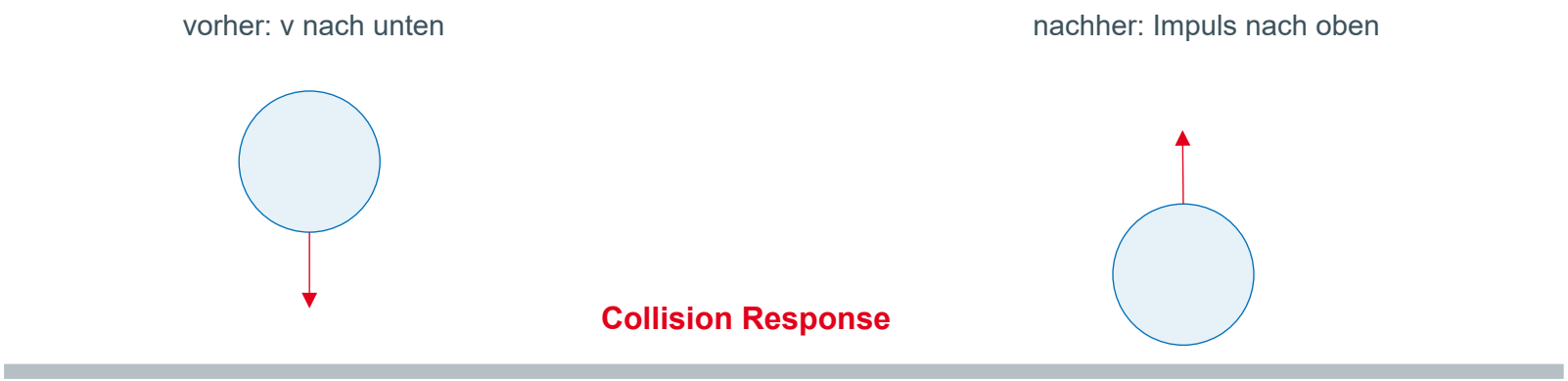
- Impulserhaltung beschreibt, dass der Gesamtimpuls in einem abgeschlossenen System erhalten bleibt
- Kollisionen werden mit Impulsen beschrieben

Formeln:

- Impuls eines Körpers: $\mathrm{p} = m * v$
- Vor der Kollision gilt: p_{vor}
- Nach der Kollision gilt: p_{nach}
- Grundidee: $p_{vor} = p_{nach}$

Impulserhaltung erklärt, warum Impulse in Kollisionen verwendet werden: Der Gesamtimpuls vor und nach dem Stoß bleibt erhalten

Beispiel: Ball trifft Boden



Restitution entscheidet, ob der Ball nur stoppt oder sichtbar zurückspringt.

Relative Velocity

- Bestimmt ob und wie stark eine Kollision stattfindet
- $v_{rel} = (v_B - v_A) * n$
- $v_{rel} < 0 \rightarrow$ Körper bewegen sich aufeinander zu
- $v_{rel} > 0 \rightarrow$ Körper entfernen sich voneinander
- $v_{rel} = 0 \rightarrow$ Kein relativer Normalanteil der Geschwindigkeit

v_A = Velocity of body A at Contact

v_B = Velocity of body B at Contact

n = Normal from A to B

Impulsberechnung

Wie Groß muss der Impuls sein?

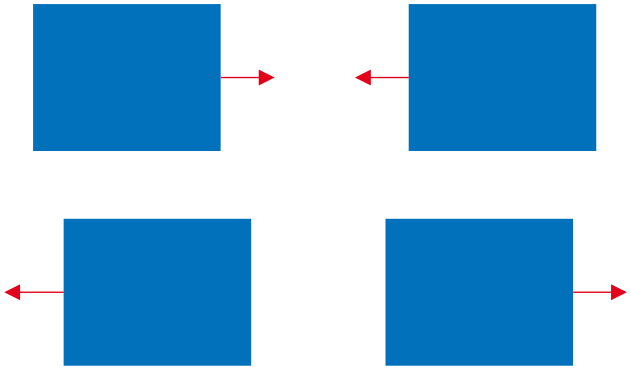
$$j = -\frac{(1+e)(v_r * n)}{\frac{1}{m_A} + \frac{1}{m_B}}$$

- e = Restitution
- v_r = relative Geschwindigkeit
- n = Kontakt-Normale

Kontaktimpuls anwenden

$$v'_A = v_A - \frac{J}{m_A}$$

$$v'_B = v_B + \frac{J}{m_B}$$

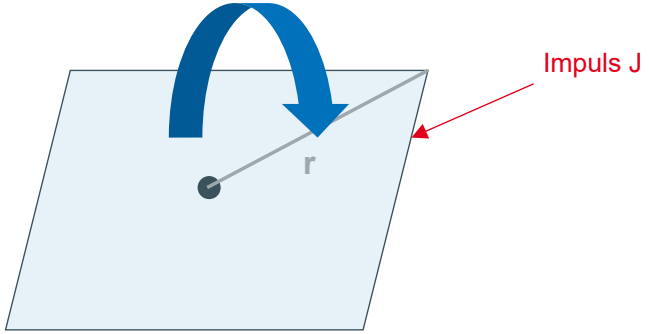


Der Kontaktimpuls wird auf beide Körper mit entgegengesetztem Vorzeichen angewendet und ändert so ihre Geschwindigkeiten

Impuls wirkt auf Rotation

Wenn der Impuls nicht durch den Schwerpunkt geht:

- Lineare Geschwindigkeit ändert sich
- Winkelgeschwindigkeit ändert sich
- Körper beginnt zu rotieren



Ein Kontaktimpuls hat deshalb lineare und angulare Effekte.

Impuls wirkt auf Rotation mathematisch

$$\boldsymbol{\tau = r * J}$$

$\tau = Drehmoment$
$r = Hebelarm$
$J = Impuls\ (Vektor)$

$$\boldsymbol{\Delta w = I^{-1}(r * J)}$$

$\Delta w = Winkelgeschwindigkeitsänderung$
$I = Inertiatensor\ (Trägheitstensor)$

Effective Mass

$$m_{eff} = \left(\frac{1}{m} + (r * n)^T I^{-1} (r * n)\right)^{-1}$$

- Effective mass beschreibt die Geschwindigkeitsänderung pro Impuls in einer bestimmten Richtung.
- Sie hängt nicht nur von der Masse ab.
- Auch Kontaktpunkt, Trägheit und Rotation spielen eine Rolle.

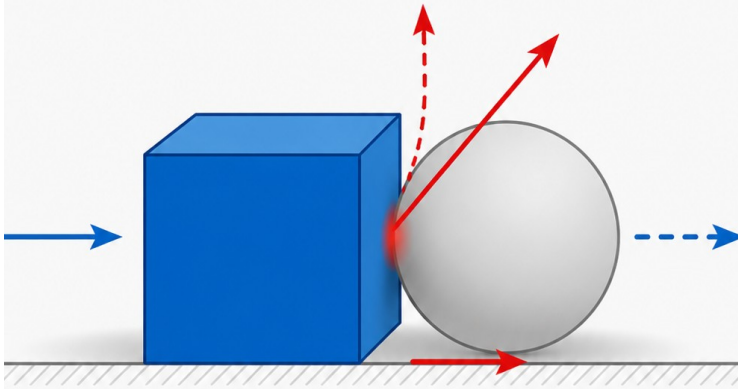
leicht
großes Δv

schwer
kleines Δv

off-center
Rotation

6

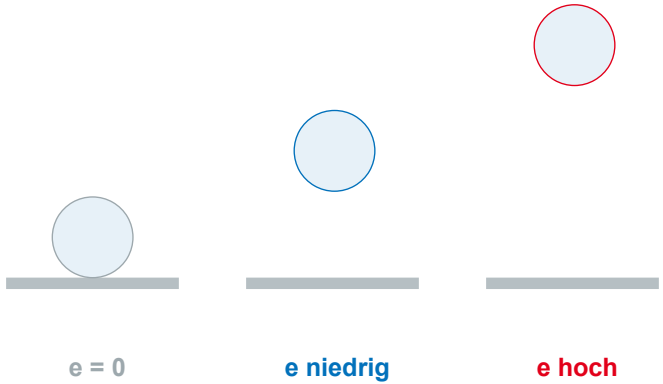
Contact Constraints, Restitution, Friction



Restitution: Bounciness

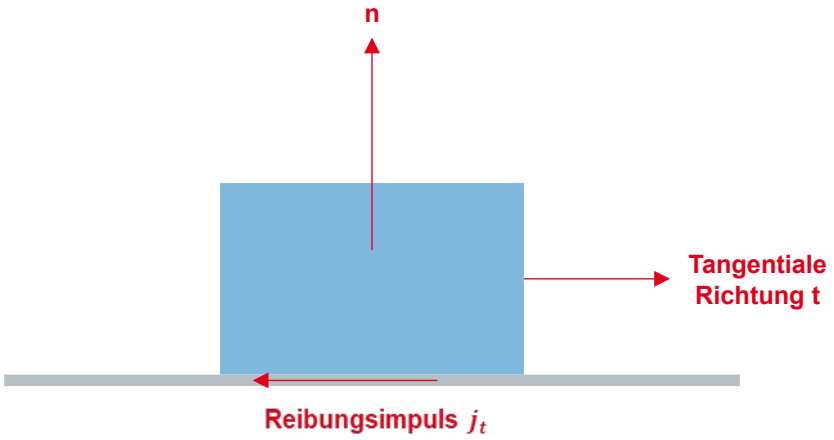
Restitution	Verhalten
0	Kein Abprallen
niedrig	Körper springt kaum
hoch	Körper springt stark zurück

- Je größer die Restitution e, desto stärker prallt ein Körper zurück



Friction

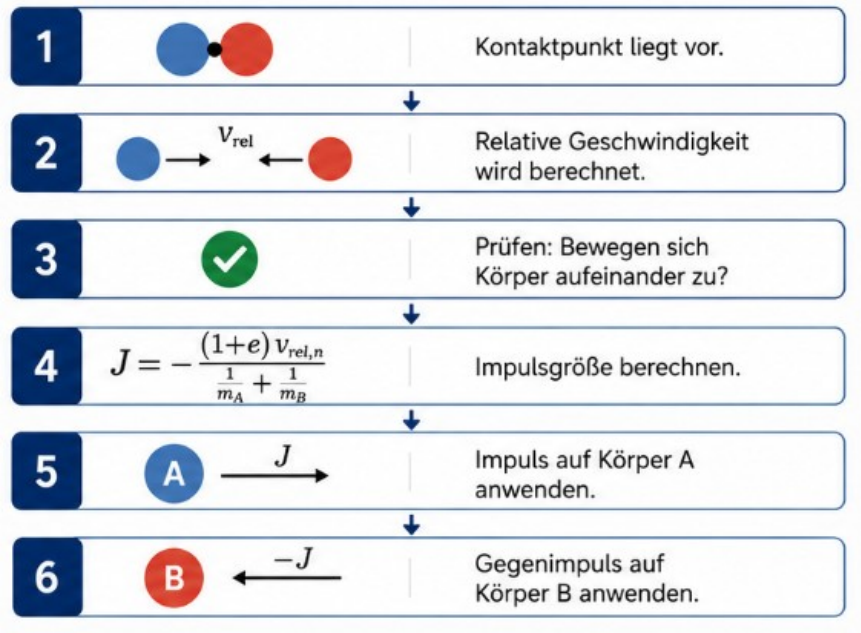
- Reibung wirkt tangential zur Kontaktfläche
- Reduziert Rutschen und Beeinflusst Stapelstabilität
- Wichtig für Fahrzeuge, Charaktere, Kisten
- Coulomb-Grenze: $|j_t| \leq \mu * j_n$



Single Collision Solver

Vereinfachter Ablauf:

1. Kontaktpunkt liegt vor
2. Relative Geschwindigkeit wird berechnet.
3. Prüfen: Bewegen sich Körper aufeinander zu?
4. Impulsgröße berechnen.
5. Impuls auf Körper A anwenden.
6. Gegenimpuls auf Körper B anwenden.



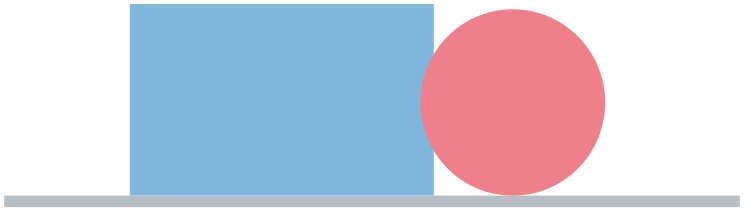
Penetration Correction

Problem:

- Diskrete Zeitschritte führen zu Eindringung
- Körper können leicht ineinander liegen -> Korrektur nötig

Methoden:

- Baumgarte Stabilization
- Split Impulse
- Coordinate Projection



Sequential Impulse Solver

```
for iteration = 1..10
        für jeden Kontakt:
                relative velocity berechnen
                Impuls berechnen
                Impuls anwenden
```

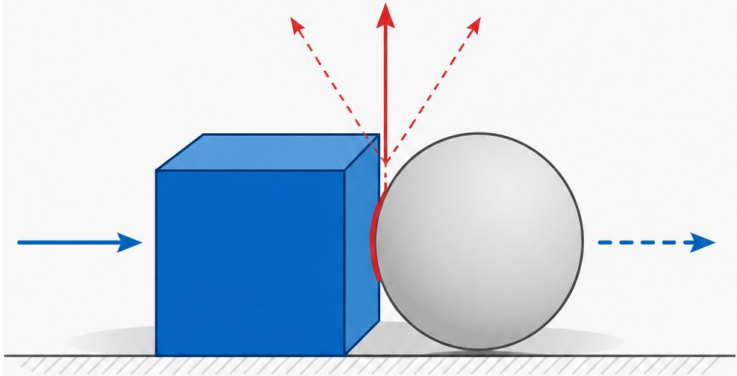
- Bullet benutzt keine exakten Solver
- Kontakte werden iterativ gelöst
- Mehr Iterationen = höhere genauigkeit

Bullet Physics Pipeline

1. applyGravity() - Kräfte Anwenden
2. integrateVelocities() - Geschwindigkeiten aktualisieren
3. collisionDetection() - Kollisionen erkennen
4. generateContacts() - Kontaktpunkte erzeugen
5. Solve Constraints() - Kontakte / Impulse lösen
6. integrateTransforms() - Positionen und Rotation aktualisieren

7

pybullet Demo



Zusammenfassend / Ausblick

1. Keine Perfekte Realität
2. Kollisionen werden über kurze Impulse gelöst
3. Besserer Solver für komplexe Kontakte
4. Mehr Bedeutung durch VR, Robotik und realistischere Spielewelten

Quellen

- Erwin Coumans: Bullet Physics Simulation - Introduction to rigid body dynamics and collision detection, SIGGRAPH 2015.
- Bullet Physics SDK / bullet3: Dokumentation und Example Browser.
- pybullet Quickstart Guide: Python-API, GUI, Simulation Steps, URDF-Beispiele.
- Erin Catto: Numerical Methods und Constraint Solving in Game Physics.
- Gino van den Bergen: Collision Detection in Interactive 3D Environments.
- OpenAI / ChatGPT: KI-generierte Bilder zu Impuls, Kollision, Reibung und starren Körpern. Erstellt durch eigene Prompts für diese Präsentation, 2026.