23.4.2023

Physic Engines Lab

Teil 2: Beschleunigung und unelastischer Stoss

Mohammad Schazad (mohamsch) & Sevici Alperen (sevicalp)

ZHAW

Inhaltsverzeichnis

1 Zusammenfassung

Dieses Semesterprojekt simuliert zwei Würfel, die durch zwei Stösse mit unterschiedlichen Kräften beeinflusst werden. Der Bericht beschreibt den Versuchsaufbau in Unity und die Physik hinter den beiden Stössen. Ausserdem werden Position, Geschwindigkeit und Impuls der beiden Würfel sowie der Gesamtimpuls als Funktion der Zeit grafisch dargestellt. Der erste Würfel stösst gegen eine Feder an der Wand und wird in die entgegengesetzte Richtung geschleudert. Dabei kommt es zur Kollision mit einem zweiten Würfel, an dem der erste Würfen an den zweiten Würfel haftet.

2 Aufbau des Experiments

Die Aufgabe besteht darin, einen Versuch mit zwei Würfeln durchzuführen und die physikalischen Vorgänge zu beschreiben. Der erste Würfel wird mit einer konstanten Kraft auf eine Geschwindigkeit von $2\frac{m}{s}$ beschleunigt. Der Würfel 1 hat eine Masse von $2kg$ und eine Kantenlänge von $1.5m$. Die Beschleunigungsphase soll beendet sein, bevor der Würfel 1 die Wand erreicht.

Danach soll der Würfel 1 mit einer Feder elastisch gestossen werden, wobei die Federkonstante und die Länge der Feder so zu wählen sind, dass der Würfel nicht gegen die Wand stösst und der Stoss langsam erfolgt. Die Energie des Würfels und der zusammengedrückten Feder wird für die Berechnung verwendet.

Würfel 1 gleitet nach dem elastischen Stoss ohne Reibung nach links und stösst dann unelastisch mit Würfel 2 zusammen. Würfel 2 hat die gleiche Grösse und Masse wie Würfel 1. Der Aufprall soll mittels Unitys eingebaute Funktion «OnCollision» implementiert und die beiden Würfel sollen mit einem «Fixed Joint» sicher miteinander verbunden werden.

Das Experiment wird in einem geeigneten Raum durchgeführt, der ausreichend Platz für den Bewegungsbereich der Würfel bietet. Der Versuchsaufbau sollte so gestaltet sein, dass die Bewegung der Würfel gut beobachtet und gemessen werden kann.

3 Physikalische Beschreibung der einzelnen Vorgänge

3.1 Beschleunigung Würfel 1

Im ersten Teil des Versuchs wird der Würfel 1 im Zeitpunkt $t_0 = 0$ und mit einer Geschwindigkeit von $v_0 = 0$ aus der Ruhe mit einer konstanten Kraft beschleunigt. Mit der Annahme, dass der Würfel 1 auf eine Geschwindigkeit von $v_1 = 2\frac{m}{s}$ mit einer Beschleunigungszeit von $\Delta t = 2s$ kommen soll, sieht die Beschleunigung a wie folgt aus:

$$a = \frac{\Delta v}{\Delta t} = \frac{(v_1 - v_0)}{(t - t_0)} = \frac{2\frac{m}{s} - 0\frac{m}{s}}{2s - 0s} = 1\frac{m}{s^2}$$

(Da wir uns in diesem Experiment grundsätzlich nur auf der x-Achse bewegen, wird im gesamten Bericht für die Definition der Formeln auf die Vektorschreibweise verzichtet.)

Mit der berechneten Beschleunigung kann nun mit dem zweiten Newton'schen Axiom (Aktionsprinzip) die Kraft F, die sich auf den Würfel wirken soll, berechnet werden. Der Würfel 1 hat eine Masse von $m_1 = 2kg$ und mit der vorberechneten Beschleunigung $a = 1\frac{m}{s^2}$ sieht die Kraft F wie folgt aus:

$$F = m_1 * a = 2kg * 1\frac{m}{s^2} = 2N$$

3.2 Elastischer Stoss Würfel 1 mit der Feder

Im zweiten Teil des Versuchs stösst der Würfel elastisch gegen eine Wand. Die Federkonstante k und die Länge der Feder l müssen so gewählt werden, dass der Würfel nicht gegen die Wand stösst und der Stoss langsam erfolgt.

Die kinetische Energie des Würfels $E_{kinWürfel}$ vor dem Stoss wird in die potenzielle Energie der Feder $E_{potFeder}$ und dann in die kinetische Energie des Würfels nach dem Stoss umgewandelt. Mit der Annahme, dass der Stoss vollkommen elastisch wirkt und wir uns in einem abgeschlossenen System befinden, kann davon ausgegangen werden, dass die Gesamtenergie aufgrund des Energieerhaltungssatz gleich bleibt. Somit gilt:

$$E_{kinWürfel} = \frac{1}{2}mv^2 = \frac{1}{2}kx^2 = E_{potFeder}$$

Die Geschwindigkeit v_1 und der Impuls p_1 des Würfels ändern sich während des Stosses, aber der Gesamtimpuls p_{gesamt} bleibt gleich:

$$\sum p_{gesamt\_vor\_Stoss} = m_1 * v_1 + m_2 * v_2 = m'_1 * v'_1 + m'_2 * v'_2 = \sum p_{gesamt\_nach\_Stoss}$$

3.3 Unelastischer Stoss Würfel 1 mit Würfel 2

Nach dem Stoss gleitet der Würfel 1 reibungslos nach links und stösst unelastisch mit dem Würfel 2 zusammen. Die beiden Würfel werden nach dem Prall durch ein festes Gelenk verbunden und bewegen sich ab dann als ein System.

Beim unelastischen Stoss wird ein Teil der kinetischen Energie in andere Energieformen, sei es Deformation, Schall oder Wärme, umgewandelt. Der Gesamtimpuls p_{gesamt} des Systems bleibt jedoch erhalten:

$$p_{gesamt} = p_1 + p_2 = (m_1 * v_1) + (m_2 * v_2)$$

Da beide Würfel durch ein festes Gelenk verbunden sind und davon ausgegangen werden kann, dass die aneinander haften bleiben, ist die Geschwindigkeit beider Würfel nach dem unelastischen Stoss gleich und es wird mit der Geschwindigkeit des Schwerpunkts v_{Spt} vom gesamten System weitergerechnet. Um die Geschwindigkeit des Schwerpunkts v_{Spt} zu berechnen, kann die Impulserhaltungsgleichung wie folgt verwendet werden:

$$p_{gesamt} = (m_1 + m_2) * v_{Spt}$$

$$v_{Spt} = \frac{p_{gesamt}}{(m_1 + m_2)}$$

Da Würfel 1 und Würfel 2 die gleiche Masse m besitzen und zudem Würfel 2 vor dem Stoss sich in Ruhe befindet $v_2 = 0$ ergibt sich:

$$v_{Spt} = \frac{m_1 * v_1}{(m_1 + m_2)} = \frac{2kg * v_1}{2kg + 2kg} = \frac{1}{2} * v_1$$

Was schlussfolgernd bedeutet, dass sich die beiden Würfel mit der Hälfte der anfänglichen Geschwindigkeit von Würfel 1 weiterbewegen.

4 Beschreibung der Implementierung

Es wurden drei verschiedene C#-Codes geschrieben, um erstens den Würfel zu beschleunigen. Der zweite Code dient zur Konfiguration der Feder und der dritte Code zur Messung der Beschleunigung. Diese Implementierungen werden weiter unten erklärt.

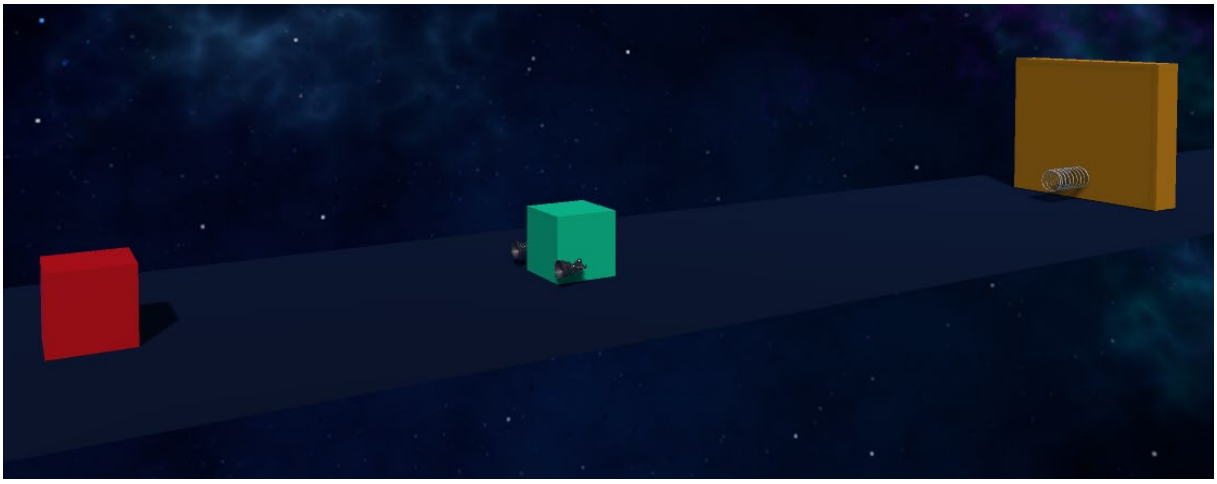

Abbildung 1: Startansicht des Aufbaus in Unity

4.1 Elemente

Im Folgenden werden die einzelnen Elemente und ihre Eigenschaften aufgezeigt.

4.1.1 Würfel 1

Der Würfel 1 hat eine Seitenlänge von 1.5m und ein Gewicht von 2kg. Zusätzlich sind zwei Düsen an den Seiten des Würfels angehängt, welche während des Experiments in der Beschleunigungsphase brennen und somit die Beschleunigung visualisieren.

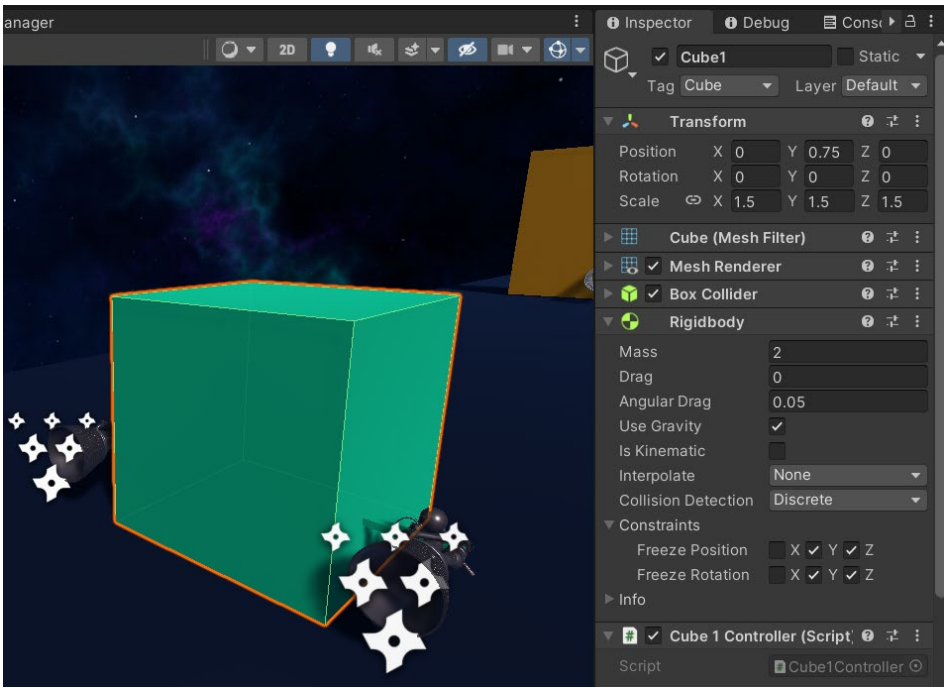

Abbildung 2: Würfel 1 Eigenschaften

4.1.2 Würfel 2

Der Würfel 2 hat die gleichen Dimensionen und Masse wie der Würfel 1.

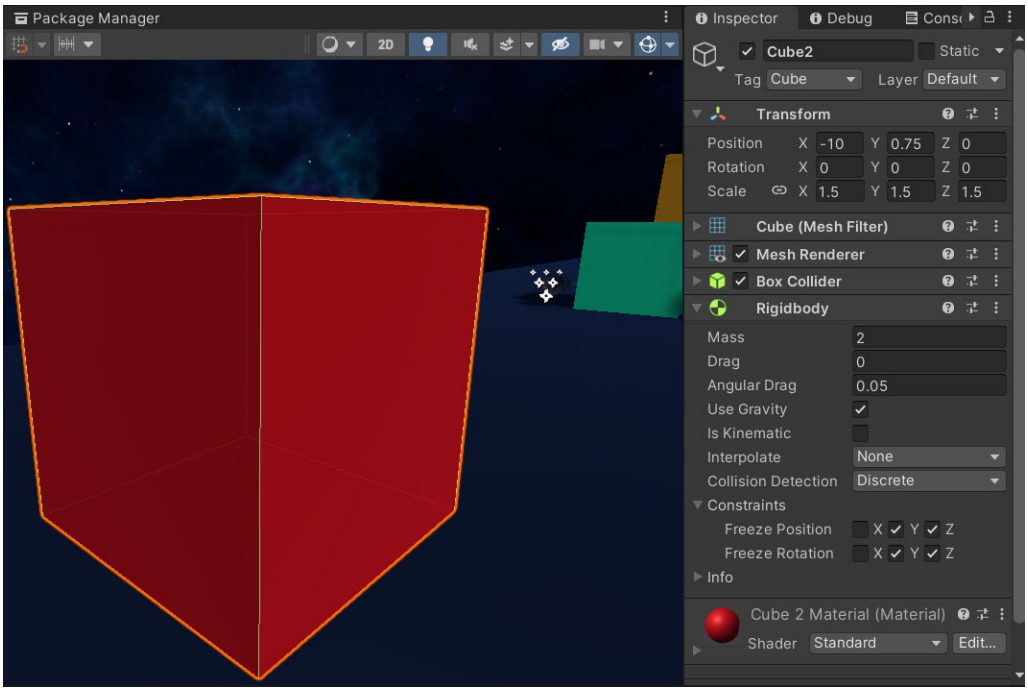

Abbildung 3: Würfel 2 Eigenschaften

4.1.3 Wand/Feder

Die Wand hat ein Gewicht von 100kg, damit beim Stoss die Wand sich nicht bewegt. Der «BoxCollider» ist die Umrandung der Wand, in welche die Kollision mit dem Würfel 1 detektiert wird. Zusätzlich befindet sich innerhalb des «Colliders» eine Feder, die während der Federung komprimiert und dekomprimiert wird und somit den elastischen Stoss visualisieren soll.

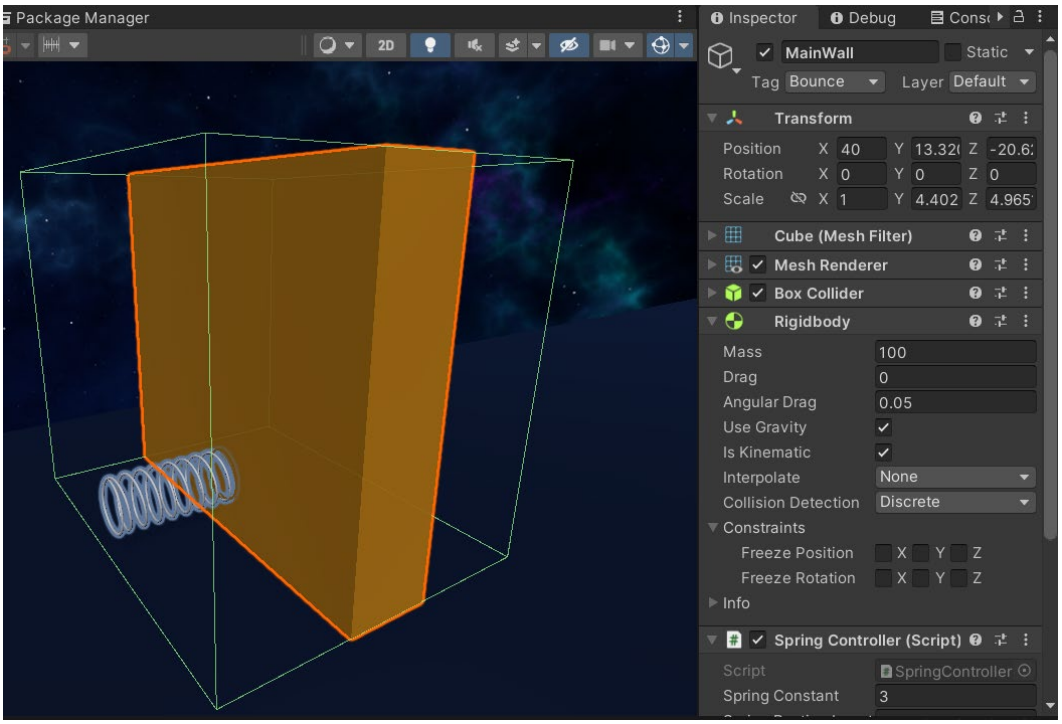

Abbildung 4: Wand mit Feder

4.2 Beschleunigung Würfel 1

Zu Beginn wurden verschiedene Eigenschaften des Würfels festgelegt. Dazu gehören die Zielgeschwindigkeit des Würfels von $2\frac{m}{s}$ und die Zeit, die er braucht, um seine Zielgeschwindigkeit von $2\frac{m}{s}$ zu erreichen, wurde auf 2 Sekunden definiert.

Beim Starten fangen die Düsen an zu brennen und der Würfel 1 fängt an zu beschleunigen und gleitet damit reibungslos in x-Richtung bis zur Wand und wird danach durch die Feder zurück gestossen in die entgegengesetzte Richtung, so dass er mit Würfel 2 kollidiert und angeheftet wird.

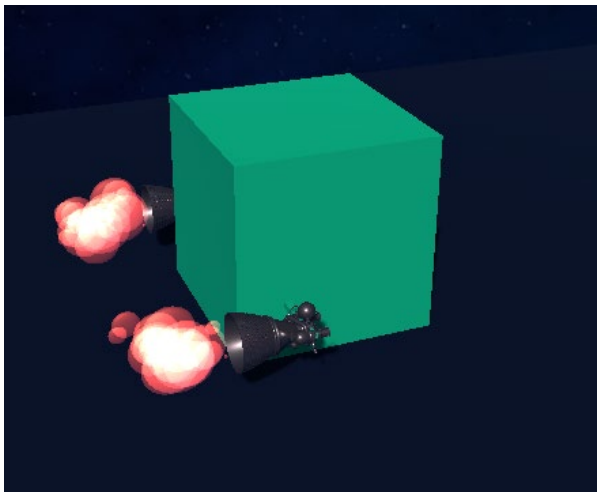

Abbildung 5: Beschleunigung Würfel 1

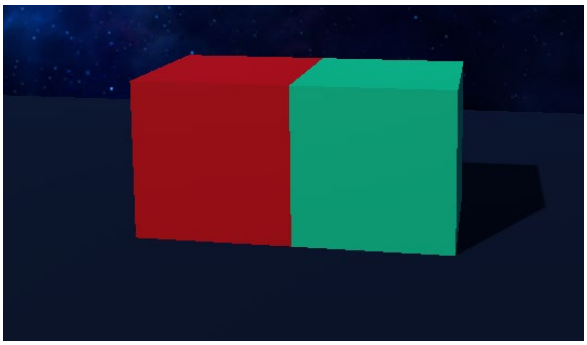

Abbildung 6: Kollision der beiden Würfeln

4.3 Elastischer Stoss

Nach der Beschleunigung gleitet der Würfel 1 eine Zeit lang bis er die Wand erreicht. Wie im Kapitel 3.2 erläutert, wird der Würfel zuerst aufgrund der Federkraft bis zum Stillstand abgebremst und dann wieder auf die gewünschte Eintrittsgeschwindigkeit in entgegengesetzte Richtung beschleunigt.

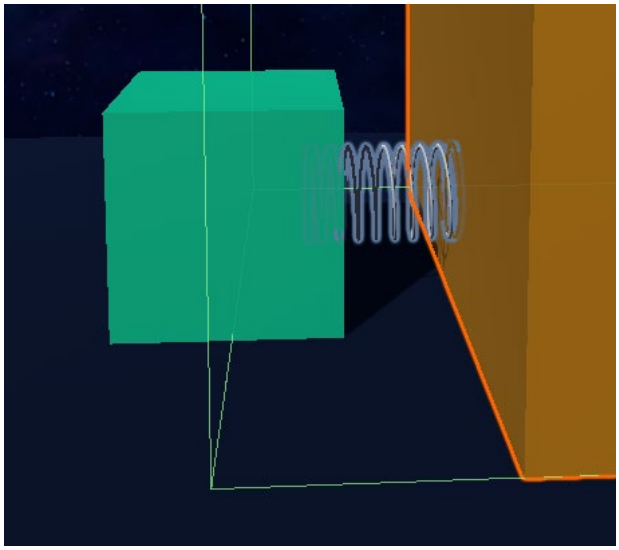

Abbildung 7: Federung des Würfels 1

4.4 Datensammlung

Während des Experiments in Unity werden die Werte Zeit t, Position vom Würfel 1 $x_1(t)$, Position vom Würfel 2 $x_2(t)$, Geschwindigkeit vom Würfel 1 $v_1(t)$, Geschwindigkeit vom Würfel 2 $v_2(t)$, Impuls vom Würfel 1 $p_1(t)$, Impuls vom Würfel 2 $p_2(t)$ festgehalten und nach Beenden des Experiments mittels WriteTimeSeriesToCSV() in eine CSV-Datei geschrieben. Aus diesen Daten werden danach grafische Darstellungen erzeugt, welche im Kapitel 5 beschrieben werden.

5 Resultate mit grafischer Darstellung

Im Ortsdiagramm wird die Position der beiden Würfel dargestellt. Man sieht, wie Würfel 1 die Richtung wechselt und sich nach Zusammenstoss mit Würfel 2 gemeinsam in negativ linear fortbewegt.

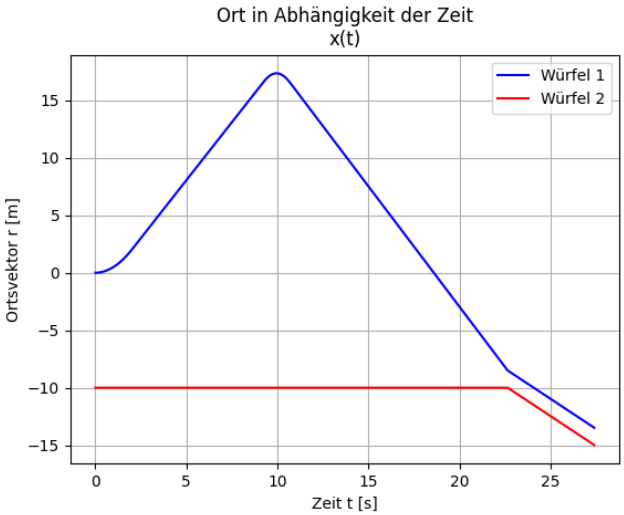

Diagramm 1: Ort als Funktion der Zeit von beiden Würfeln

Im Geschwindigkeitsdiagramm sieht man, dass Würfel 1 innerhalb von 2 Sekunden auf $2\frac{m}{s}$ beschleunigt und eine Zeit lang mit gleicher Geschwindigkeit sich fortbewegt. Bis die Feder angestossen wird und nach der maximalen Auslenkung der Feder in entgegengesetzte Richtung beschleunigt wird. Würfel 2 bleibt stehen, bis er mit Würfel 1 kollidiert und mit einem Stoss auf Geschwindigkeit $1\frac{m}{s}$ erhöht und Würfel 1 auf $1\frac{m}{s}$ reduziert wird.

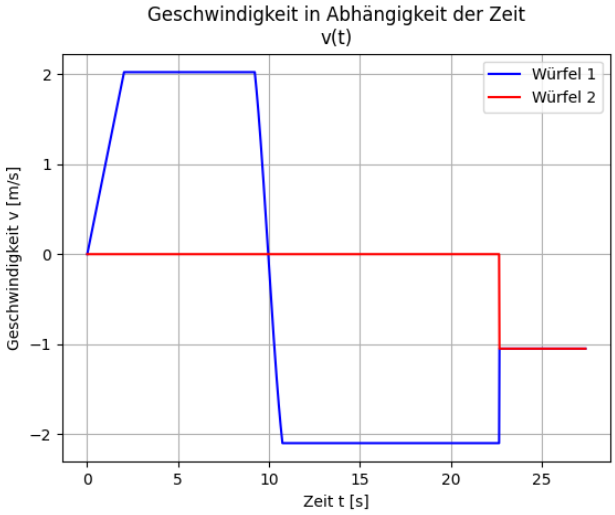

Diagramm 2: Geschwindigkeit als Funktion der Zeit von beiden Würfeln

Im Impulsdiagramm sieh man ein ähnliches Verhalten wie im Geschwindigkeitsdiagramm. Der Impuls des Würfels 1 ist das Doppelte ihrer Geschwindigkeit. Beim Zusammenstoss der beiden Würfel, wird dieser somit von 4 auf 2 reduziert, da sich die Masse verdoppelt.

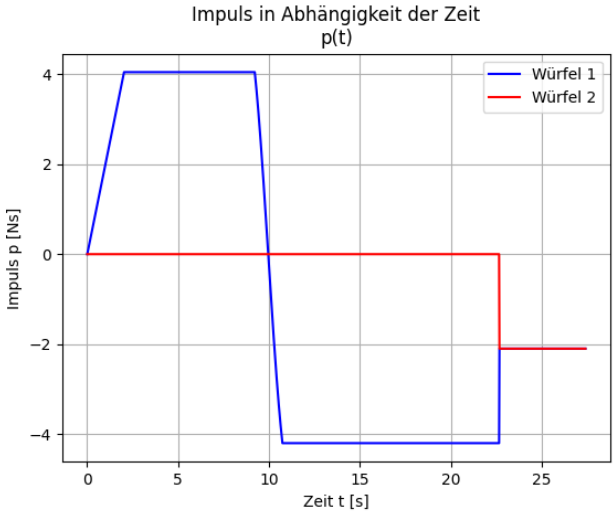

Diagramm 3: Impuls als Funktion der Zeit von beiden Würfeln

6 Rückblick und Lehren aus dem Versuch (nach Teil 3)

7 Quellenverzeichnis

Es wurden keine Quellen benutzt, sondern nach dem Wissen und Beobachtung der Autoren geschrieben.

7.1 Abbildungsverzeichnis

7.2 Diagrammverzeichnis

8 Anhang C#-Code

8.1 Würfel 1

```
using UnityEngine;

public class Cube1Controller : MonoBehaviour
{
    private Rigidbody rb;
    private float targetVelocity; // m/s
    private float accelerationTime; // s
    private float calculatedForce; // N
    private float calculatedAcceleration; // m/s
    private bool reachedTargetSpeed;

    void Start()
    {
        rb = GetComponent<Rigidbody>();
        targetVelocity = 2f;
        accelerationTime = 2f;
        calculatedAcceleration = targetVelocity / accelerationTime;
        calculatedForce = rb.mass * calculatedAcceleration;
        reachedTargetSpeed = false;
    }

    void FixedUpdate()
    {
        if (rb.velocity.x < targetVelocity && !reachedTargetSpeed)
        {
            rb.AddForce(calculatedForce, 0, 0);
            return;
        }
        reachedTargetSpeed = true;
    }

    private void OnCollisionEnter(Collision other)
    {
        if (other.gameObject.CompareTag("Cube"))
        {
            gameObject.AddComponent<FixedJoint>();
            gameObject.GetComponent<FixedJoint>().connectedBody =
other.rigidbody;
        }
    }
```

```
}
```

8.2 Wand/Feder

```
using UnityEngine;

public class SpringController : MonoBehaviour
{
    public float springConstant;
    public float springRestingLength;
    public float compressionSpeed;
    private float springForce;
    private Rigidbody rb;
    private Rigidbody collidingCubRb;
    private bool isCompressed;

    // animation
    public GameObject springAnimatedGO;
    private float timer = 0;
    private float animationSpeed = .09f;


    private void Start()
    {
        // initialLength = 5f;
        // springConstant = 1000f;
        // compressionSpeed = .1f;
        rb = GetComponent<Rigidbody>();
    }

    private void OnTriggerEnter(Collider collider)
    {
        if (!collider.attachedRigidbody) return;
        collidingCubRb = collider.attachedRigidbody;
        isCompressed = true;
    }

    private void OnTriggerExit(Collider collider)
    {
        if (collider.attachedRigidbody)
        {
            isCompressed = false;
        }
    }
```

```
    private void FixedUpdate()
    {
        if (!isCompressed) return;
        Vector3 velocity = collidingCubRb.velocity;
        float posXDelta = Mathf.Abs(transform.position.x - colli-
dingCubRb.position.x);
        float compressionLength = springRestingLength - posXDelta;
        springForce = -springConstant * compressionLength;
        collidingCubRb.AddForce(springForce, 0, 0);

        if (velocity.x > 0)
        {
            animateSpring(-compressionSpeed, 3 * compressionSpeed);
        }
        else
        {
            animateSpring(compressionSpeed, -3 * compressionSpeed);
        }
    }

    private void animateSpring(float scale, float pos)
    {
        if (timer < animationSpeed)
        {
            timer += Time.deltaTime;
        }
        else
        {
            springAnimatedGO.transform.localScale += new Vec-
tor3(scale, 0, 0);
            springAnimatedGO.transform.position += new Vector3(pos,
0, 0);
            timer = 0;
        }
    }
}
```

8.3 Dantesammlung

```
using UnityEngine;
using System.Collections.Generic;
using System.IO;

public class StatsLoggerController : MonoBehaviour
{
    public Rigidbody c_1;
    public Rigidbody c_2;
    private float currentTimeStep; // s
    private float currentVelocityX; // m/s
    private float currentImpulseX; // N*s
    private float currentForce; // N
    private List<List<float>> timeSeries;

    void Start()
    {
        timeSeries = new List<List<float>>();
    }

    void FixedUpdate()
    {
        currentTimeStep += Time.deltaTime;
        timeSeries.Add(new List<float>() {
            currentTimeStep,
            c_1.position.x,
            c_2.position.x,
            c_1.velocity.x,
            c_2.velocity.x,
            c_1.velocity.x * c_1.mass,
            c_2.velocity.x * c_2.mass,
            (c_1.velocity.x / currentTimeStep) * c_1.mass, //
F=a*m=(v/deltaT)*m
            (c_2.velocity.x / currentTimeStep) * c_2.mass,
        });
    }

    void OnApplicationQuit()
    {
        // WriteTimeSeriesToCSV();
    }

    void WriteTimeSeriesToCSV()
```

```
    {
        using (var streamWriter = new StreamWriter("time_se-
ries.csv"))
        {
            streamWriter.Write-
Line("t,x1(t),x2(t),v1(t),v2(t),p1(t),p2(t),F1(t),F2(t)");
            foreach (List<float> timeStep in timeSeries)
            {
                streamWriter.WriteLine(string.Join(",", timeStep));
                streamWriter.Flush();
            }
        }
    }
}
```