

Semesterprojekt Modul PE FS 2023

TEIL 2

BADR OUTITI (OUTITBAD), ARMANDO SHALA (SHALAAR3), SILVAN BAACH (BAACHSIL)

Inhaltsverzeichnis

1.	<u>ZUSAMMENFASSUNG</u>	2
2.	<u>AUFBAU DES EXPERIMENTS</u>	2
3.	<u>PHYSIKALISCHE BESCHREIBUNG DER EINZELNEN VORGÄNGE</u>	2
3.1.	HINFAHRT	2
3.2.	ELASTISCHE KOLLISION	3
3.3.	RÜCKFAHRT	4
3.4.	INELASTISCHE KOLLISION	4
4.	<u>BESCHREIBUNG DER IMPLEMENTIERUNG INKL. SCREENSHOTS AUS UNITY (VERSUCHSAUFBAU)</u>	5
4.1.	EIGENSCHAFTEN WÜRFEL 1	5
4.2.	EIGENSCHAFTEN WÜRFEL 2	5
4.3.	EIGENSCHAFTEN FLÜSSIGKEITSRAKETENANTRIEB	6
4.4.	EIGENSCHAFTEN FLAMME	6
4.5.	EIGENSCHAFTEN WAND	7
4.6.	EIGENSCHAFTEN FEDER	7
4.7.	EIGENSCHAFTEN UMGEBUNG	8
4.8.	EIGENSCHAFTEN TV UND TISCH	8
5.	<u>RESULTATE MIT GRAFISCHER DARSTELLUNG GEMÄSS DER DETAILLIERTEN AUFGABENSTELLUNG</u>	9
5.1.	GESCHWINDIGKEITSGESTALTUNG WÜRFEL 1	9
5.2.	IMPULS WÜRFEL 1	9
5.3.	WEG WÜRFEL 1	9
5.4.	GESCHWINDIGKEITSGESTALTUNG WÜRFEL 2	10
5.5.	IMPULS WÜRFEL 2	10
5.6.	WEG WÜRFEL 2	10
5.7.	IMPULS BEIDER WÜRFEL	11
6.	<u>RÜCKBLICK UND LEHREN AUS DEM VERSUCH (NACH TEIL 3)</u>	11
7.	<u>QUELLENVERZEICHNIS</u>	11
7.1.	QUELLENANGABEN	11
7.2.	ABBILDUNGSVERZEICHNIS	11
7.3.	BILDER UND GRAFIKEN	11
7.4.	3D MODELLE	12
8.	<u>C#-CODE</u>	12

1. Zusammenfassung

Würfel 1, gekennzeichnet durch das Logo unseres Bronze-Sponsors, wird durch eine konstante Beschleunigung welche reibungsfrei verläuft aus dem Stillstand auf eine Geschwindigkeit von ca. 2 m/s gebracht. Die Beschleunigungsphase endet vor dem elastischen Aufprall mit der Wand, an dem eine Feder montiert ist. Der Würfel wird von der Feder konstant und langsam abgebremst. Da das Experiment reibungsfrei durchgeführt wird und der Würfel die Wand nicht berührt, wird die gesamte Energie, welche die Feder aufnimmt, wieder an den Würfel 1 zurückgegeben und er gleitet reibungsfrei in die entgegengesetzte Richtung. Würfel 1 stösst anschliessend inelastisch mit Würfel 2 zusammen. Sobald die Würfel von der Plane herunterfallen, ist das Experiment abgeschlossen.

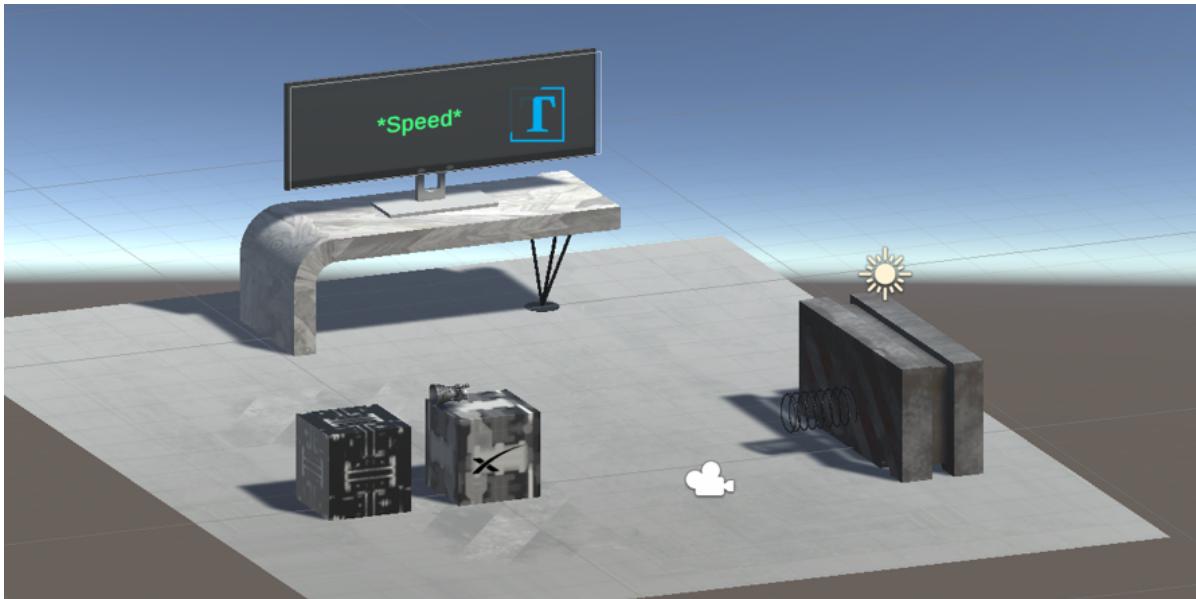


Abbildung 1: Ausgangslage des Experiments

2. Aufbau des Experiments

In diesem Experiment werden Beschleunigung und inelastischer Stoß untersucht. Ein Würfel von 2kg und 1.5m Seitenlänge wird durch eine konstante Kraft beschleunigt und erreicht in wenigen Sekunden 2 m/s, bevor er auf eine Feder an eine Wand trifft. Der Würfel kollidiert elastisch mit der Wand unter Verwendung einer geeigneten, vorgeschalteten Feder, sodass der Stoß langsam abläuft und ohne Berührung mit der Wan. Anschliessend gleitet Würfel 1 reibungsfrei zurück und stösst inelastisch mit einem identischen Würfel 2 zusammen. Der Zusammenstoß wird mittels OnCollision und einem festen Gelenk modelliert. Das Experiment analysiert die Anordnung sowie die physikalischen Prozesse und stellt Ort, Geschwindigkeit, Impuls beider Würfel und Gesamtimpuls grafisch dar.

3. Physikalische Beschreibung der einzelnen Vorgänge

3.1. Hinfahrt

In der ersten Phase der Aufgabe bewegt sich Würfel 1 (der Rechte Würfel in Abbildung 1) unter dem Einfluss einer konstanten Kraft nach rechts, bis er auf die Feder trifft. Die Masse beider Würfel beträgt 2 kg und die Seitenlänge beträgt 1.5 m. Die konstante Kraft ist so dimensioniert, dass Würfel 1 nach wenigen Sekunden eine Geschwindigkeit von ungefähr 2 m/s erreicht.

Zur Berechnung der Beschleunigung des Würfels wird das zweite Newtonsche Gesetz angewendet. Mit Kenntnis der Beschleunigung und der Masse des Würfels lässt sich die konstante Kraft F bestimmen:

$$F = m * a$$

ermitteln, indem die gewünschte Endgeschwindigkeit v und die Beschleunigungszeit t berücksichtigt werden:

$$v = a * t$$

Hierbei steht F für die konstante Kraft, m für die Masse des Würfels und a für die Beschleunigung. Die Kraft F lässt sich Daraus folgt:

$$a = \frac{v}{t}$$

In dieser Phase bewegt sich Würfel 1 unter dem Einfluss der konstanten Kraft F horizontal nach rechts. Die Beschleunigungsphase endet, bevor Würfel 1 die Feder und der Wand erreicht. Anschliessend erfolgt eine elastische Kollision des Würfels mit der Feder an der Wand.

Die Federkonstante k und die Länge der Feder sind so bemessen, dass Würfel 1 nicht in die Wand prallt und dass der Stoss langsam abläuft. Zur Sicherstellung dieser Bedingungen wird die Energie vom Würfel und der komprimierter Feder herangezogen. Die potenzielle Energie der komprimierten Feder entspricht der kinetischen Energie des Würfels:

$$\frac{1}{2} * k * x^2 = \frac{1}{2} * m * v^2$$

Hierbei steht x für die Kompression der Feder. Aus dieser Gleichung lässt sich die Federkonstante k bestimmen, um die gewünschten Bedingungen für den elastischen Stoss zu erfüllen.

Zusammenfassend besteht die erste Phase der Aufgabenstellung im physikalischen Vorgang der Beschleunigung von Würfel 1 durch eine konstante Kraft, bis dieser seine Zielgeschwindigkeit erreicht. Dabei werden die Gleichungen des zweiten Newtonschen Gesetzes und der Energieerhaltung herangezogen, um die erforderlichen Grössen wie Kraft, Beschleunigung und Federkonstante zu berechnen.

3.2. Elastische Kollision

In der nächsten Phase der Aufgabenstellung kommt es zum elastischen Stoss von Würfel 1 mit der Feder. Die Feder, die an der Wand befestigt ist, dient als Stoßdämpfer und ermöglicht eine elastische Kollision, bei der die kinetische Energie und der Impuls nach und vor dem Stoss konserviert bleiben. Die potenzielle Energie der komprimierten Feder entspricht dabei der kinetischen Energie des Würfels vor dem Stoss:

$$\frac{1}{2} * k * x^2 = \frac{1}{2} * m * v^2$$

Hierbei ist k die Federkonstante, x die Kompression der Feder, m die Masse von Würfel 1 und v seine Geschwindigkeit vor dem Stoss. Aus dieser Gleichung lässt sich die Federkonstante k in Abhängigkeit von der gewünschten Kompression x ermitteln.

$$k = m * v^2 * x^2$$

Während des elastischen Stosses wirkt die Federkraft entgegen der Bewegungsrichtung von Würfel 1 und verlangsamt dessen Geschwindigkeit. Die Federkraft kann durch das Hookesche Gesetz beschrieben werden:

$$F = -k * x$$

Dabei ist F die Federkraft und x die momentane Kompression der Feder. Das negative Vorzeichen impliziert, dass die Kraft entgegengesetzt zur Kompressionsrichtung wirkt. Da der Stoss elastisch ist, bleibt der Impuls von Würfel 1 erhalten. Der Impuls p lässt sich wie folgt berechnen:

$$p = m * v$$

Da die Masse m konstant ist, hängt der Impuls direkt von der Geschwindigkeit v ab. Nach dem elastischen Stoss gleitet Würfel 1 reibungsfrei nach links zurück.

Zusammenfassend wird in dieser Phase der Aufgabenstellung der elastische Stoss von Würfel 1 mit der Feder untersucht. Dabei wird das Hookesche Gesetz und das Prinzip der Impulserhaltung angewendet, um die Federkraft und den Impuls während des Stossvorgangs zu analysieren. Der elastische Stoss führt dazu, dass Würfel 1 seine Bewegungsrichtung ändert und reibungsfrei zurückgleitet.

3.3. Rückfahrt

Nach dem elastischen Stoss mit der Feder beginnt eine neue Phase, in der Würfel 1 reibungsfrei nach links gleitet. In dieser Phase ist die kinetische Energie des Würfels konserviert, da keine externen Kräfte auf ihn wirken und keine Energie in Form von Reibung verloren geht. Da die Geschwindigkeit von Würfel 1 unmittelbar nach dem elastischen Stoss bekannt ist, kann die Bewegung des Würfels während dieser Phase beschrieben werden. Die Position des Würfels als Funktion der Zeit lässt sich durch die Gleichung für gleichförmige Bewegung ausdrücken:

$$x(t) = x_0 + v * t$$

Dabei ist $x(t)$ die Position des Würfels zum Zeitpunkt t , x_0 die Anfangsposition und v die Geschwindigkeit des Würfels nach dem elastischen Stoss.

Da Würfel 1 reibungsfrei gleitet, bleibt sein Impuls während dieser Phase ebenfalls erhalten. Der Impuls p kann wie folgt berechnet werden:

$$p = m * v$$

Hierbei ist m die Masse von Würfel 1 und v seine Geschwindigkeit nach dem elastischen Stoss.

Während dieser Phase nähert sich Würfel 1 kontinuierlich Würfel 2 an, der sich in einem ruhenden Zustand befindet. Beide Würfel haben die gleiche Masse und Grösse. Die Phase endet kurz bevor es zum inelastischen Stoss zwischen Würfel 1 und Würfel 2 kommt.

Zusammenfassend beschreibt diese Phase die Bewegung von Würfel 1 nach dem elastischen Stoss mit der Feder bis kurz vor dem inelastischen Stoss mit Würfel 2. Während dieser Phase gleitet Würfel 1 reibungsfrei nach links, wobei seine kinetische Energie und sein Impuls erhalten bleiben. Die Position des Würfels in dieser Phase kann mithilfe der Gleichung für gleichförmige Bewegung als Funktion der Zeit ausgedrückt werden.

3.4. Inelastische Kollision

In der letzten Phase kommt es zum inelastischen Stoss zwischen Würfel 1 und Würfel 2. Bei einem inelastischen Stoss haften die beteiligten Objekte nach der Kollision aneinander, und ein Teil der kinetischen Energie wird in innere Energie umgewandelt. Während des inelastischen Stosses bleibt der Gesamtimpuls der beiden Würfel erhalten. Um den inelastischen Stoss zu modellieren wird die `OnCollision()` Funktion verwendet um die beiden Würfel mit einem Fixed Joint sicher zusammenzuheften. Dies stellt sicher, dass die Würfel nach dem Stoss gemeinsam weitergleiten. Da der Gesamtimpuls der beiden Würfel erhalten bleibt, kann die Geschwindigkeit der beiden Würfel nach dem inelastischen Stoss berechnet werden. Der Impulserhaltungssatz für diese Situation lautet:

$$m_1 * v_{1\text{initial}} + m_2 * v_{2\text{initial}} = (m_1 + m_2) * v_{\text{final}}$$

Hierbei sind m_1 und m_2 die Massen von Würfel 1 und Würfel 2, $v_{1\text{initial}}$ und $v_{2\text{initial}}$ sind ihre Geschwindigkeiten vor dem Stoss, und v_{final} ist die gemeinsame Geschwindigkeit der beiden Würfel nach dem inelastischen Stoss. Da Würfel 2 in Ruhe ist, beträgt $v_{2\text{initial}}$ 0. Nach dem inelastischen Stoss bewegen sich die beiden Würfel mit der gemeinsamen Geschwindigkeit v_{final} reibungsfrei weiter.

Zusammenfassend behandelt diese Phase den inelastischen Stoss zwischen Würfel 1 und Würfel 2. Bei diesem Stoss bleibt der Gesamtimpuls der beiden Würfel erhalten, während ein Teil der kinetischen Energie in innere Energie umgewandelt wird. Nach dem Stoss haften beide Würfel aneinander und gleiten gemeinsam mit der gleichen Geschwindigkeit weiter, die durch den Impulserhaltungssatz berechnet werden kann.

4. Beschreibung der Implementierung inkl. Screenshots aus Unity (Versuchsaufbau)

Entgegen der Vorgabe der Kapiteleinteilung des Berichts, ist die Auflistung des Versuchs unerlässlich für dessen Kredibilität. Bei Experimenten werden Beobachtungen in kontrollierten Umgebungen, wie Laboren oder Entwicklungsumgebungen durchgeführt, um möglichst viele Faktoren zu steuern. Eine Kontrolle kann durch das Weglassen oder Verwenden eines bekannten Werts für gewisse Faktoren erfolgen. Experimente müssen wiederholbar und überprüfbar sein und genau protokolliert werden, damit andere Wissenschaftler dieselben Ergebnisse erzielen.

4.1. Eigenschaften Würfel 1

Würfel 1 hat eine Masse von 2 Kilogramm und eine Seitenlänge von 1.5 Meter. Weder die Farbgebung, das Sponsorlogo noch der Flüssigkeitsraketenantrieb inkl. Flamme besitzen eine Masse. Die Existenz dieser genannten Elemente beschränkt sich ausschliesslich auf Design entscheide. Der Körper an sich besteht aus reibungslosem Material, welcher auch nur in einer kontrollierten, künstlichen Umgebung existiert.

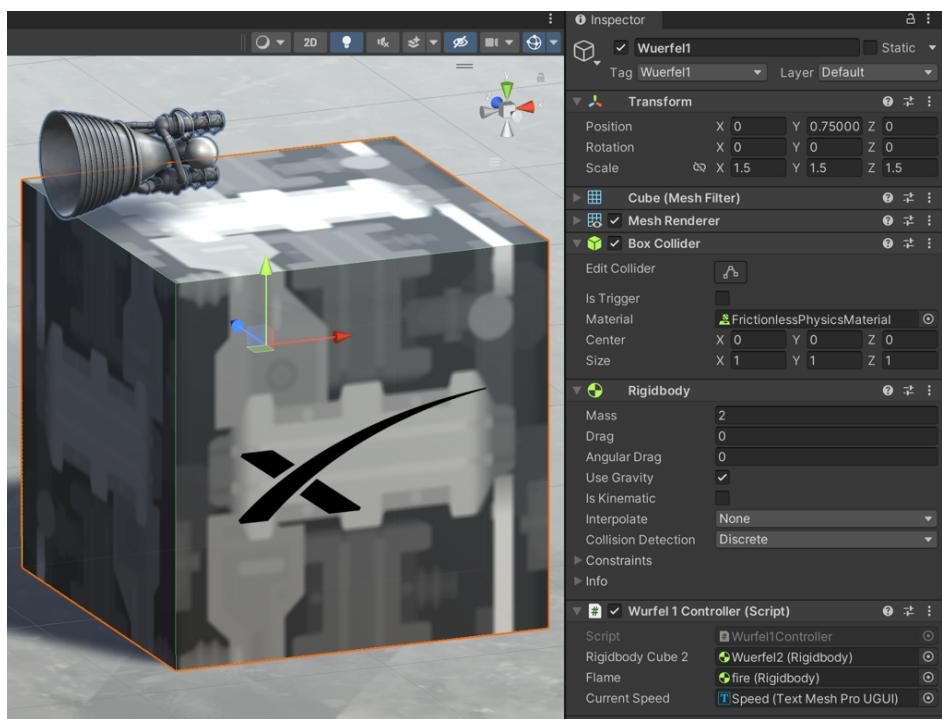


Abbildung 2: Eigenschaften Würfel 1

4.2. Eigenschaften Würfel 2

Würfel 2 ist identisch zu Würfel 1, ausser dem fehlendem Flüssigkeitsraketenantrieb.

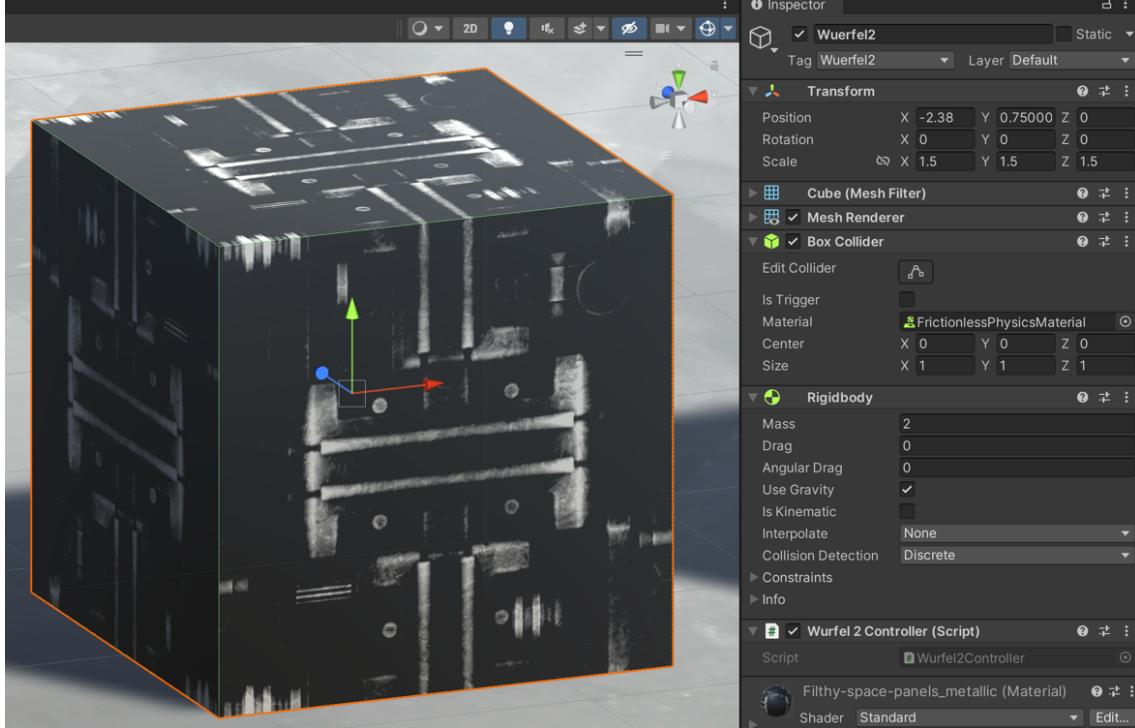


Abbildung 3: Eigenschaften Würfel 2

4.3. Eigenschaften Flüssigkeitsraketenantrieb

In Realität hat der Flüssigkeitsraketenantrieb nicht nur eine Masse, sondern auch einen genügend grossen Tank inkl. Kabel und Schläuche, welche die Funktionalität sicherstellen. Da dies aber keinerlei Relevanz für dieses Experiment ausweist, wurde der Masse und jegliche anderen Bauteile keinerlei Aufmerksamkeit geschenkt. Für eine realitätsnahe Funktionsweise der Komponenten verweisen wir auf unsere hauseigene [Customer Success Abteilung](#).

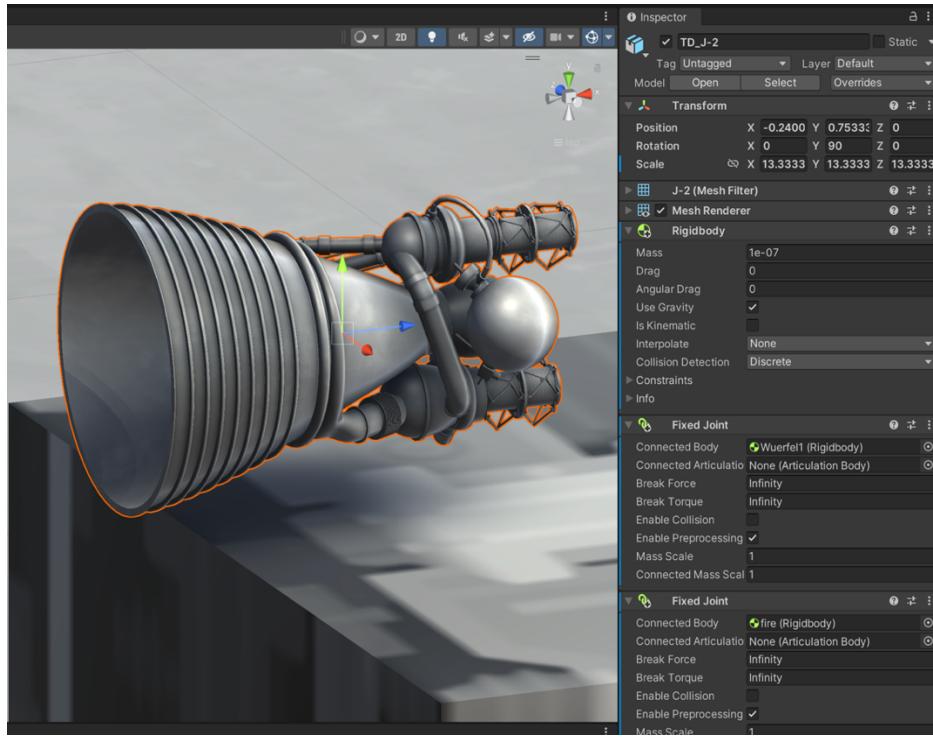


Abbildung 4: Eigenschaften Flüssigkeitsraketenantrieb

4.4. Eigenschaften Flamme

Auch hier wurde die Masse auf eine vernachlässigbare Grösse gesetzt. Dies ist auf die zuvor erwähnte kontrollierte Umgebung zurückzuführen.

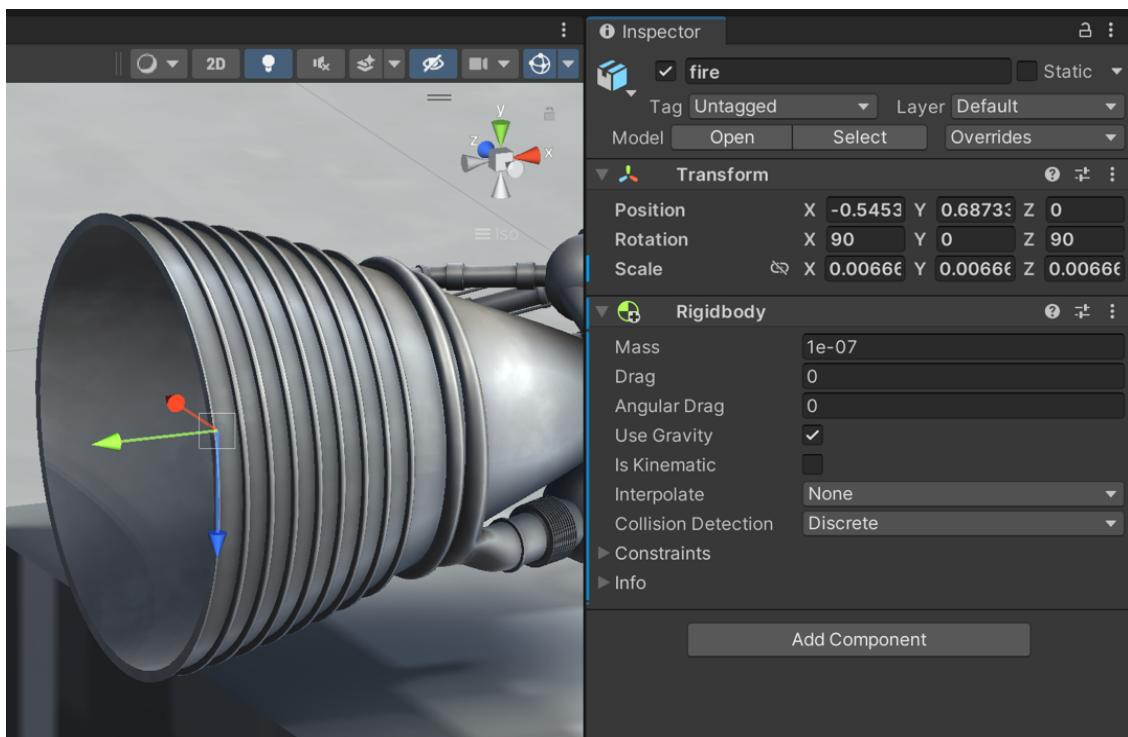


Abbildung 5: Eigenschaften Flamme

4.5. Eigenschaften Wand

Die Wand dient als Halterung der Feder. Der Versuch beinhaltet keinen Berührungs punkt der Wand mit Würfel 1, da die Feder beim Aufprall mit Würfel 1 zusammengedrückt wird und dies somit verhindert.

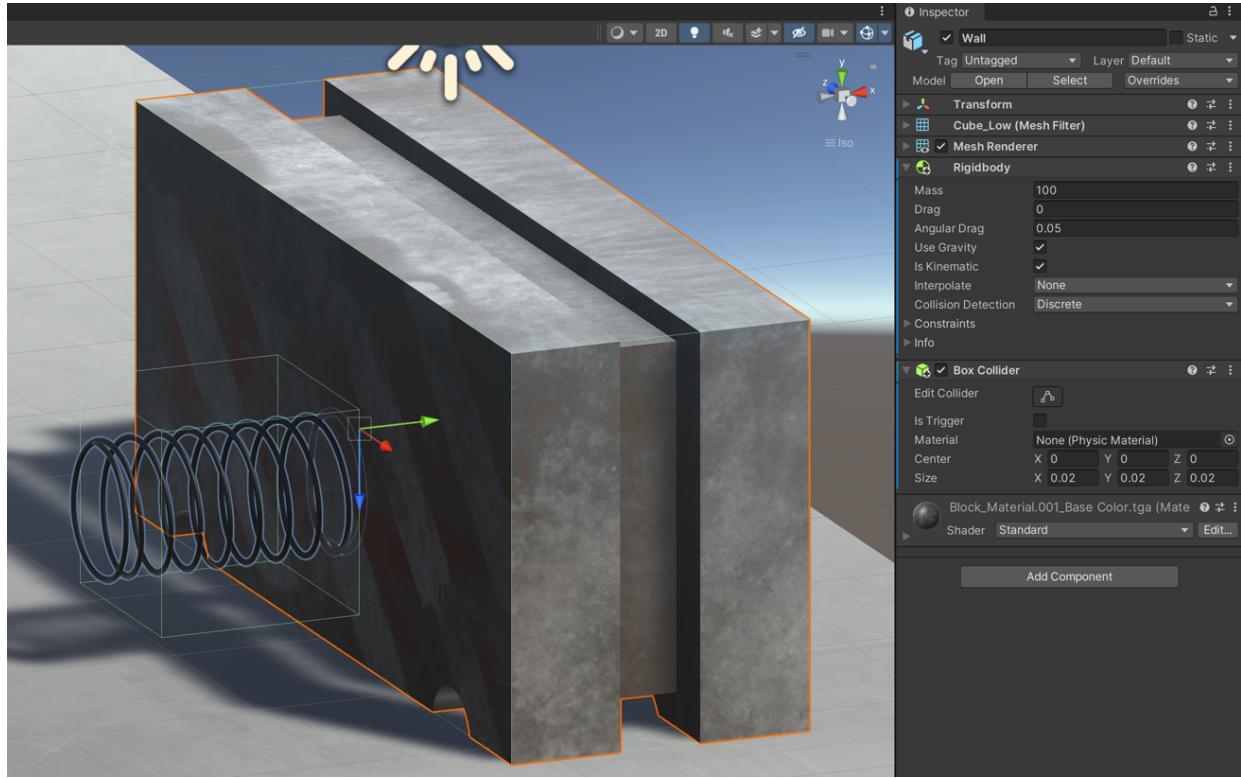


Abbildung 6: Eigenschaften Wand

4.6. Eigenschaften Feder

In diesem Experiment dient die Feder der optischen Aufbesserung sowie dem Verständnis des Experiments. Eine visuelle Stauchung ist mit dem jetzigen Projektstand nicht umgesetzt. Weitere Eigenschaften, wie Federkonstante der Feder sind innerhalb des Codes gesetzt. Diese befinden sich im Kapitel der Codes, welcher im hinteren Teil des Projektes heimisch ist.

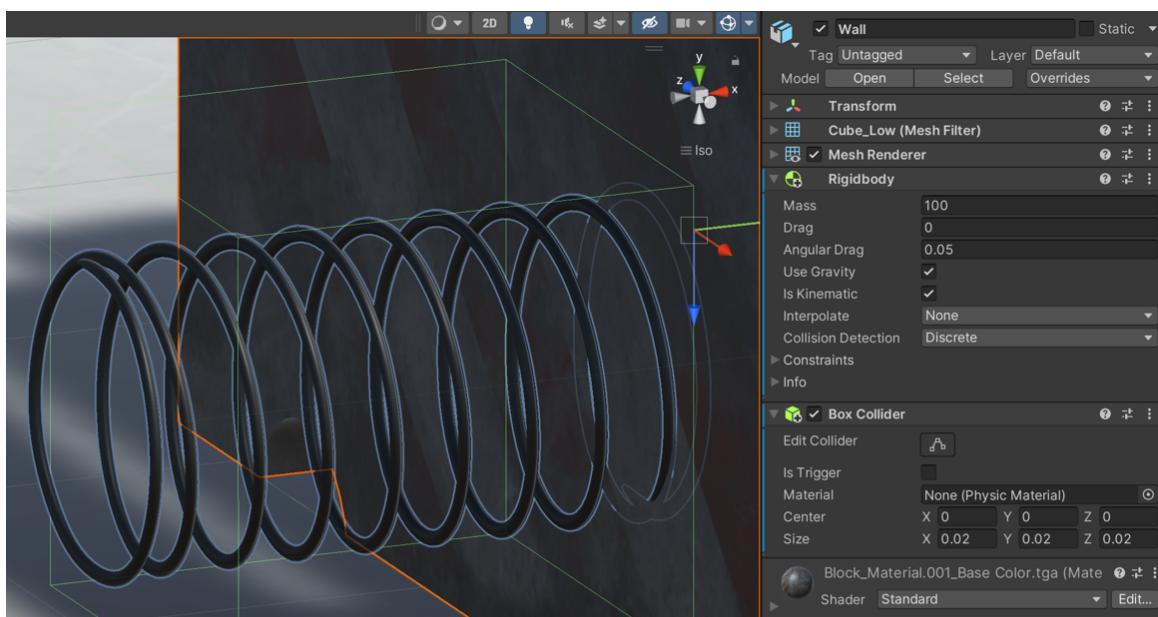


Abbildung 7: Eigenschaften Feder

4.7. Eigenschaften Umgebung

Die Umgebung des Versuchs obliegt der kontrollierten Umgebung des Versuchsaufbaus. Relevante physikalische Beziehungen, wie Gravitationsanziehung und Kollisionseinwirkung, werden von Unity gehandhabt und deshalb wird nicht weiter auf diesen Aspekt eingegangen. Zudem sind alle Arten von Wetterbedingungen nicht simuliert. In der kontrollierten Umgebung durchbrechen steht Sonnenstrahlen aus der Nachmittagszeit den wolkenfreien Himmel.

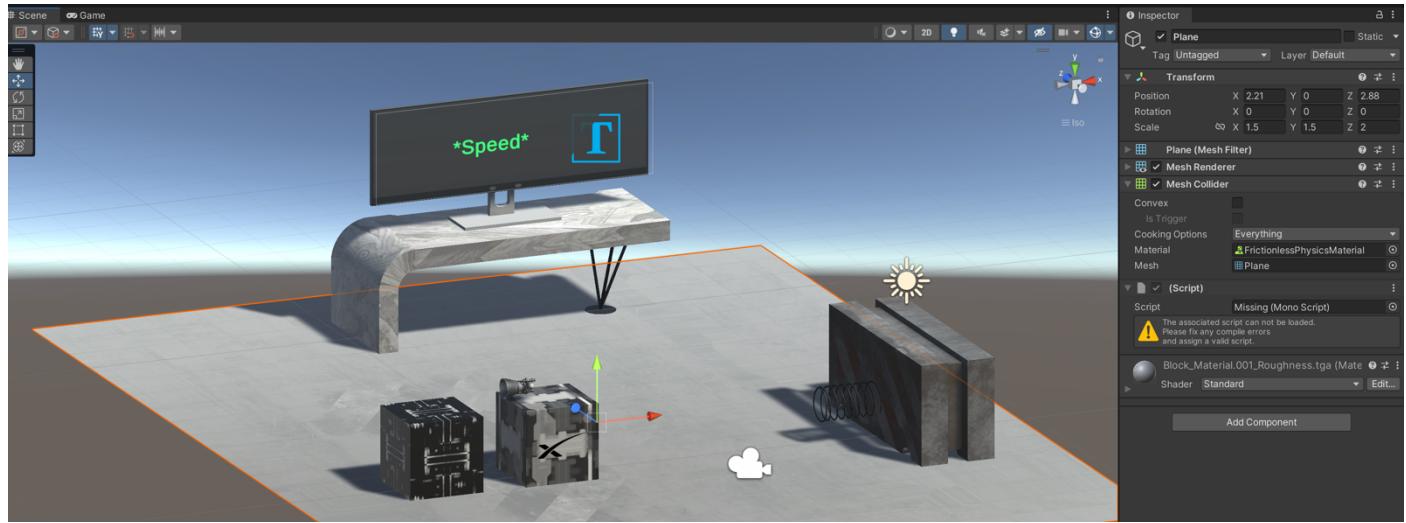


Abbildung 8: Eigenschaften Umgebung

4.8. Eigenschaften TV und Tisch

Die Existenzberechtigung dieser Objekte konzentriert sich auf einen gemeinsamen Punkt. Der Text, welche die aktuelle Geschwindigkeit von Würfel 1 zu jedem beliebigen Zeitpunkt während der Laufzeit repräsentiert, wird auf einem Monitor dargestellt, welcher selbst auf einem Tisch steht.

5. Resultate mit grafischer Darstellung gemäss der detaillierten Aufgabenstellung

5.1. Geschwindigkeitsgestaltung Würfel 1

Würfel 1 wird sowohl reibungsfrei als konstant mit der Formel $v_x = a_{x0} * t + v_{x0}$ beschleunigt und erreicht seine Zielgeschwindigkeit nach ca. drei Sekunden, was im Diagramm durch eine lineare Steigung gefolgt von einer horizontalen Linie ersichtlich ist. Nach kurzer Zeit, in der sich die Geschwindigkeit nicht ändert, trifft der Würfel 1 elastisch auf die Feder. Die Feder wird von der Energie des Würfels zusammengestaucht und bremst diesen ab. Da das Experiment ideal ist, kommt auch der Energieerhaltungssatz zum Zuge. Entsprechend wird der Würfel mit der gleichen Energie zurückgedrückt und gleitet anschliessend reibungsfrei in die entgegengesetzte Richtung. Dies wird durch die negative Geschwindigkeit veranschaulicht. Nach der Beschleunigungsphase der Feder, welche zwischen Sekunde vier und circa sechs abgebildet ist, gleitet der Würfel so lange zurück, bis er mit Würfel 2 inelastisch kollidiert. Hierbei ist anzumerken, dass die Geschwindigkeit des Würfel 1 negativ wird, da dieser in jene Richtung gleitet, von welcher er gekommen ist. Die Kollision veranschaulicht die Geschwindigkeit mit dem positiven Ausschlag auf der Abszisse kurz vor Sekunde acht. Die Geschwindigkeit wird halbiert, da sich das Gewicht verdoppelt. Das Verhalten korrespondiert mit der Formel der Kraft $F = m * a$.

5.2. Impuls Würfel 1

Der Impuls ist eine Zusammenstellung von $m * v$ und daher besitzt er die gleiche Erklärung wie die Geschwindigkeitsgestaltung desselben Würfels. In diesem Experiment wird Würfel 1 bis maximal 2 m/s beschleunigt und hat eine Masse von 2 kg, daher ist auch die Ordinate doppelt so gross.

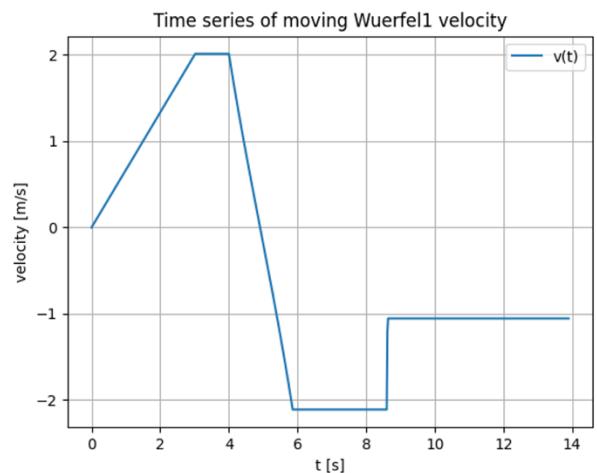


Abbildung 9: Geschwindigkeitsgestaltung Würfel 1

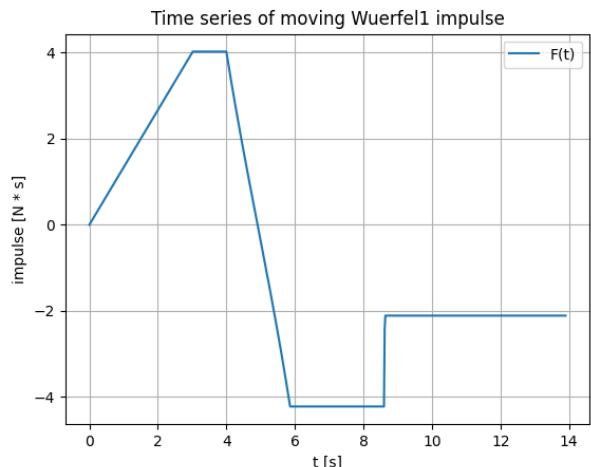


Abbildung 10: Impuls Würfel 1

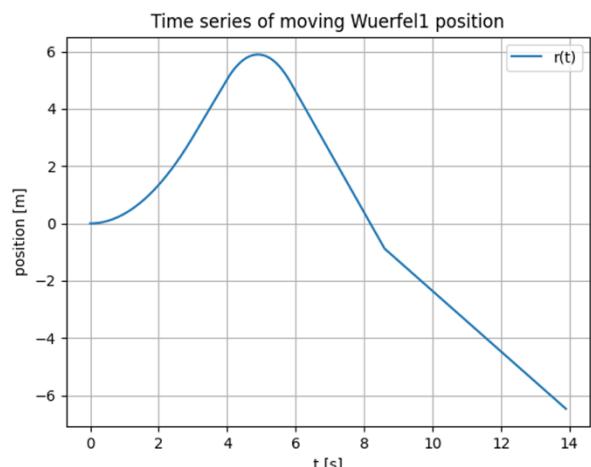


Abbildung 11: Weg Würfel 1

5.4. Geschwindigkeitsgestaltung Würfel 2

Würfel 2 hat selbst keinen Antrieb, was zu einer Ruhelage bis zu circa Sekunde sieben führt. Da schon Würfel 1 sich reibungsfrei rückwärts bewegt, reflektiert das auf Würfel 2 und demzufolge ist dessen Geschwindigkeit auch negativ. Es sieht so aus, als würde die Geschwindigkeit ruckartig auf knapp 1 m/s gesetzt. In Tatsache ist dies eine lineare Funktion mit sehr leichter Steigung. In der Grafik geht dieses Detail zugrunde, da die Skalierung zu grob für solch einen Detaillierungsgrad ist.

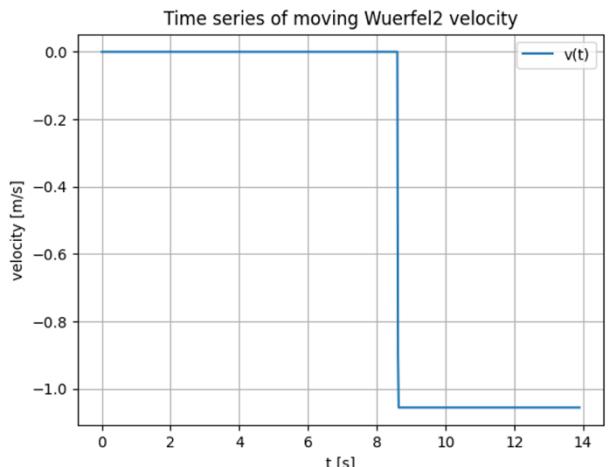


Abbildung 12: Geschwindigkeitsgestaltung Würfel 2

5.5. Impuls Würfel 2

Die Erklärung zum Impuls von Würfel 2 ist analog zu jener von Würfel 1.

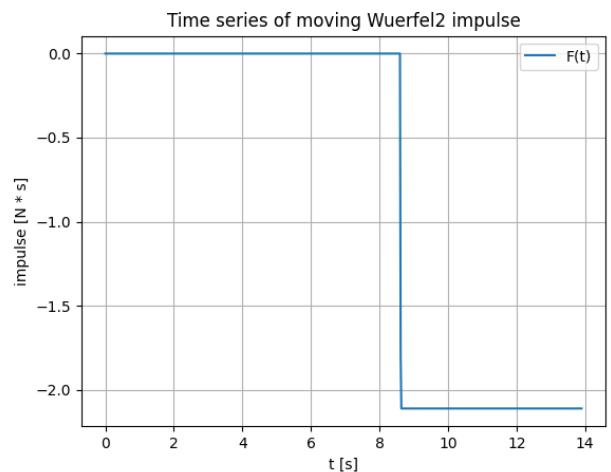


Abbildung 13: Impuls Würfel 2

5.6. Weg Würfel 2

Auch hier ist erkenntlich, dass bis in die Umgebung von Sekunde sieben die Position aufgrund mangelnder Krafteinwirkung, fundamentiert von Newtons erstem Axiom, nicht ändert. Ab dem Zeitpunkt der inelastischen Kollision gleitet Würfel 2 nach links, somit rückwärts, mit konstanter, reibungsfreier Kraft. Alle erklärten Vorgänge werden durch Beendigung der Applikation gestoppt.



Abbildung 14: Weg Würfel 2

5.7. Impuls beider Würfel

Die Grafik der kombinierten Impulse der Würfel ist, konsequenterweise, eine Symphonie der einzelnen Impulse.

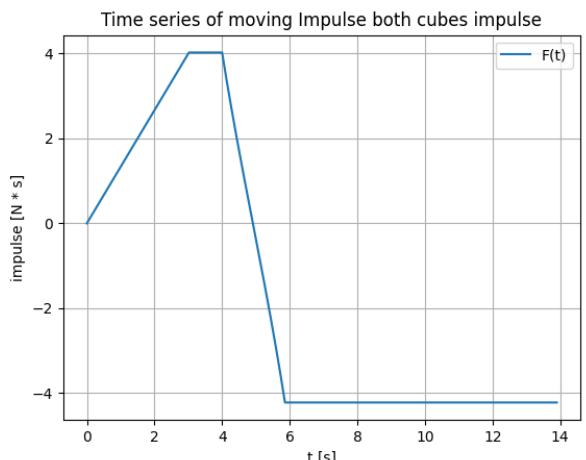


Abbildung 15: Impuls beider Würfel

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

Im vorliegenden Kapitel wird der relevante Inhalt dargelegt und erörtert, sobald dieser für den Fortschritt und das Verständnis des Experiments unerlässlich wird. Ebenso wird mit angemessenem Aufwand auf die Vernetzung mit vorherigen und nachfolgenden Kapiteln geachtet, um ein umfassendes Bild des experimentellen Zusammenhangs zu ermöglichen und eine kontinuierliche Auseinandersetzung des Berichts zu fördern.

7. Quellenverzeichnis

7.1. Quellenangaben

Alle Texte in diesem Dokument wurden ausschliesslich von den Autoren verfasst. Zusätzliche Schrift-, Bild-, Sach-, Ton- & Film- oder mündliche Quellen wurden nicht verwendet. Der Inhalt des Dokumentes basiert auf dem langjährigen Wissen der Prosaisten.

7.2. Abbildungsverzeichnis

Abbildung 1: Ausgangslage des Experiments	2
Abbildung 2: Eigenschaften Würfel 1	5
Abbildung 3: Eigenschaften Würfel 2	5
Abbildung 4: Eigenschaften Flüssigkeitsraketenantrieb	6
Abbildung 5: Eigenschaften Flamme	6
Abbildung 6: Eigenschaften Wand	7
Abbildung 7: Eigenschaften Feder	7
Abbildung 8: Eigenschaften Umgebung	8
Abbildung 9: Geschwindigkeitsgestaltung Würfel 1	9
Abbildung 10: Impuls Würfel 1	9
Abbildung 11: Weg Würfel 1	9
Abbildung 12: Geschwindigkeitsgestaltung Würfel 2	10
Abbildung 13: Impuls Würfel 2	10
Abbildung 14: Weg Würfel 2	10
Abbildung 15: Impuls beider Würfel	11

7.3. Bilder und Grafiken

Alle Bilder sowie Grafiken stammen aus eigener Produktion.

7.4. 3D Modelle

Alle 3D Elemente sowie alle Materialien sind von sketchfab.com bezogen. Lediglich Würfel 2 wurde gezeichnet. Würfel 1 wurde vom grosszügigem Dozententeam zur Verfügung gestellt.

- Sketchfab, "Concrete Table Desk," [Online]. Available: <https://sketchfab.com/3d-models/concrete-table-desk-81cdb398e2a64f52bfafec406195e437>. [Accessed: Apr. 19, 2022].
- Sketchfab, "Spiral Spring," [Online]. Available: <https://sketchfab.com/3d-models/spiral-spring-1e6a7476f88c475ab5d86d8ac11e1752>. [Accessed: Apr. 19, 2022].
- Sketchfab, "J-2S," [Online]. Available: <https://sketchfab.com/3d-models/j-2s-b579558252c4459fa624c8350443fbc0>. [Accessed: Apr. 19, 2022].
- Sketchfab, "Near Future Computer Monitor," [Online]. Available: <https://sketchfab.com/3d-models/near-future-computer-monitor-87efc29297c944c6b2dac7b30b4e2084>. [Accessed: Apr. 19, 2022].
- Sketchfab, "Bomberman Fire," [Online]. Available: <https://sketchfab.com/3d-models/bomberman-fire-8e482145eede419980fabf073fcb13c9>. [Accessed: Apr. 19, 2022].
- Sketchfab, "Concrete Block," [Online]. Available: <https://sketchfab.com/3d-models/concrete-block-11a24dbe3be24f299519387f5ffd62e>. [Accessed: Apr. 19, 2022].
- Icons.com, "SpaceX Logo Icon," [Online]. Available: https://cdn.icon-icons.com/icons2/2389/PNG/512/spacex_logo_icon_144865.png. [Accessed: Apr. 19, 2022].

8. C#-Code

Der C# sowie der Python Code wurde mit der mehrjährigen Berufserfahrung der Projektteilnehmer realisiert.

```
1  using UnityEngine;
2
3  public class CameraFollow : MonoBehaviour
4  {
5      public Transform targetObject;
6      public Transform springObject;
7      public float zoomDistanceThreshold;
8      public float zoomSpeed;
9      public float zoomFactor;
10
11     private Vector3 _initialOffset;
12     private Vector3 _cameraPosition;
13
14     // Start is called before the first frame update
15     void Start()
16     {
17         zoomDistanceThreshold = 2.0f;
18         zoomSpeed = 2.0f;
19         zoomFactor = 0.5f;
20         _initialOffset = transform.position - targetObject.position;
21     }
22
23     // Update is called once per frame
24     void FixedUpdate()
25     {
26         _cameraPosition = targetObject.position + _initialOffset;
27         var transform1 = transform;
28         transform1.position = _cameraPosition;
29
30         // Check the distance between the camera and the Spring object
31         var distanceToSpring = Vector3.Distance(transform1.position, springObject.position);
32
33         if (!(distanceToSpring < zoomDistanceThreshold)) return;
34         // Calculate a new camera position based on the distance to the Spring object
35         var position = transform.position;
36         Vector3 springDirection = (springObject.position - position).normalized;
37         float zoomAmount = (zoomDistanceThreshold - distanceToSpring) * zoomFactor;
38         Vector3 zoomPosition = position + springDirection * zoomAmount;
39
40         // Smoothly move the camera to the new zoom position
41         position = Vector3.Lerp(position, zoomPosition, Time.deltaTime * zoomSpeed);
42         transform.position = position;
43     }
44 }
```

```
1  using UnityEngine;
2
3  public class SpringController : MonoBehaviour
4  {
5      public float springConstant;
6      public float initialLength;
7      public float compressionSpeed;
8      private Rigidbody collidingRigidbody;
9      private Rigidbody _springRigidbody;
10
11     private bool _isCompressed;
12
13     private void Start()
14     {
15         springConstant = 1f;
16         compressionSpeed = 0.038f;
17         _springRigidbody = GetComponent<Rigidbody>();
18         initialLength = _springRigidbody.position.x - transform.position.x;
19     }
20
21     private void OnTriggerEnter(Collider collider)
22     {
23         if (!collider.attachedRigidbody) return;
24         collidingRigidbody = collider.attachedRigidbody;
25         _isCompressed = true;
26     }
27
28     private void OnTriggerExit(Collider collider)
29     {
30         if (!collider.attachedRigidbody) return;
31         _isCompressed = false;
32     }
33
34     private void FixedUpdate()
35     {
36         if (!_isCompressed) return;
37         var velocity = collidingRigidbody.velocity;
38         collidingRigidbody.velocity = new Vector3(velocity.x - compressionSpeed, velocity.y, velocity.z);
39         var compressionLength = initialLength - Mathf.Abs(transform.position.x - collidingRigidbody.position.x);
40         collidingRigidbody.AddForce(springConstant * compressionLength * Vector3.right);
41     }
42 }
```

```
1  using System.Collections.Generic;
2  using System.IO;
3  using UnityEngine;
4
5  namespace DefaultNamespace
6 {
7     public class Statistics
8     {
9         private readonly List<List<float>> _timeSeries = new();
10        private readonly string _objectName;
11        private readonly string _type;
12        private readonly string _typeID;
13        private readonly string _axis;
14
15
16        public Statistics(string objectName, string type, string typeID, string axis)
17        {
18            _objectName = objectName;
19            _type = type;
20            _typeID = typeID;
21            _axis = axis;
22        }
23
24
25        public void AddTimeStep(float time, float yValue)
26        {
27            _timeSeries.Add(new List<float>() {time, yValue});
28        }
29
30
31        public void WriteTimeSeriesToCsv()
32        {
33            var fileName = $"time_series_{_objectName}_{_type}_{_axis}.csv";
34            using var streamWriter = new StreamWriter(string.Format("{0}/{1}", Application.dataPath, fileName));
35            streamWriter.NewLine = null;
36            streamWriter.AutoFlush = false;
37            streamWriter.NewLine = null;
38            streamWriter.AutoFlush = false;
39            streamWriter.WriteLine($"t,{_typeID}(t)");
40            foreach (var timeStep in _timeSeries)
41            {
42                streamWriter.WriteLine(string.Join(",", timeStep));
43                streamWriter.Flush();
44            }
45        }
46    }
}
```

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 def getCsvData(pathToFile):
5     csvData = np.genfromtxt(pathToFile, delimiter=',', dtype=None, skip_header=0, encoding='utf-8')
6     return [{label: csvData[0, i], data: csvData[1:, i].astype(np.float64)} for i in range(0, csvData.shape[1])]
7
8 def plotDataOfCsv(plotData, pathToFile):
9     for i in range(1, len(plotData)):
10         plt.plot(plotData[0]['data'], plotData[i]['data'], label=plotData[i]['label'])
11
12     title = pathToFile.split('_')
13     typeOfAxis = title[3].split('.')[0]
14     axisName = f'{getUnitOfThingAndStuff(title[3])}'
15     plt.xlabel(f't [s]')
16     plt.ylabel(f'{typeOfAxis} [{axisName}]')
17     plt.title(f'Time series of moving {title[2]} {typeOfAxis}')
18     plt.legend()
19
20     plt.grid()
21     # for datatype in ['png', 'pdf', 'svg', 'eps', 'ps', 'raw', 'rgba', 'svgz', 'tiff', 'tiff', 'jpg', 'jpeg', 'pgf', 'bmp', 'gif']:
22     for datatype in ['png', 'svg']:
23         plt.savefig(f'time_series_{pathToFile}.{datatype}')
24
25     plt.show()
26
27 def getUnitOfThingAndStuff(type):
28     if type == 'impulse':
29         return 'kg * v'
30     elif type == 'position':
31         return 'm'
32     elif type == 'velocity':
33         return 'm/s'
34     else:
35         return 'no unit'
36
37
38 files = ['time_series_Wuerfell_impulse_x.csv', 'time_series_Wuerfell_position_x.csv',
39           'time_series_Wuerfell_velocity_x.csv', 'time_series_Wuerfel2_impulse_x.csv',
40           'time_series_Wuerfel2_position_x.csv', 'time_series_Wuerfel2_velocity_x.csv',
41           'time_series_Impulse both cubes_impulse_x.csv']
42 for file in files:
43     plotDataOfCsv(getCsvData(file), file)

```

```

1  using System;
2  using System.Collections.Generic;
3  using System.Globalization;
4  using DefaultNamespace;
5  using TMPro;
6  using UnityEngine;
7  using UnityEngine.Serialization;
8  using UnityEngine.UI;
9
10 public class WurfellController : MonoBehaviour
11 {
12     private Rigidbody _rigidbody;
13     public Rigidbody _rigidbodyCube2;
14
15     public Rigidbody flame;
16
17     // make new variable for TextMeshPro component
18     [FormerlySerializedAs("text")] public TextMeshProUGUI currentSpeed;
19     private float _targetSpeed;
20     private float _constantForce;
21     private float _acceleration;
22     private float _accelerationTime;
23     private bool _reachedTargetSpeed;
24     private float _springConstant;
25     private Statistics _statisticsVelocityX { get; set; }
26     private Statistics _statisticsImpulseX { get; set; }
27     private Statistics _statisticsImpulseBothCubesX { get; set; }
28     private Statistics _statisticsPositionX { get; set; }
29     private Statistics _statisticsWurfell { get; set; }
30     private Statistics _statisticsWurfel2 { get; set; }
31
32     // Start is called before the first frame update
33     void Start()
34     {
35         _rigidbody = GetComponent<Rigidbody>();
36         _targetSpeed = 2.0f;
37         _accelerationTime = 3f; // 0 < _accelerationTime < 5 seconds
38         _acceleration = _targetSpeed / _accelerationTime;
39         _constantForce = _rigidbody.mass * _acceleration;
40         _reachedTargetSpeed = false;
41         _springConstant = 0.5f;
42         _statisticsVelocityX = new(_rigidbody.name, "velocity", "v", "x");
43         _statisticsImpulseX = new(_rigidbody.name, "impulse", "F", "x");
44         _statisticsPositionX = new( rigidbody.name, "position", "r", "x");
45         _statisticsImpulseBothCubesX = new("Impulse both cubes", "impulse", "F", "x");
46     }
47
48     // FixedUpdate is called once per physics frame
49     void FixedUpdate()
50     {
51         if (_rigidbody.position.y < 0)
52         {
53             Application.Quit();
54             UnityEditor.EditorApplication.isPlaying = false;
55         }
56         _statisticsVelocityX.AddTimeStep(Time.time, _rigidbody.velocity.x);
57         _statisticsImpulseX.AddTimeStep(Time.time, _rigidbody.mass * _rigidbody.velocity.x);
58         _statisticsPositionX.AddTimeStep(Time.time, _rigidbody.position.x);
59         _statisticsImpulseBothCubesX.AddTimeStep(Time.time, _rigidbody.mass * _rigidbody.velocity.x + _rigidbodyCube2.mass * _rigidbodyCube2.velocity.x);
60
61         currentSpeed.SetText($"X-Velocity of Wurfell:\n{_rigidbody.velocity.x.ToString()}");
62         if (_rigidbody.velocity.x < _targetSpeed && !_reachedTargetSpeed)
63         {
64             _rigidbody.AddForce(_constantForce, 0, 0);
65             simulateFlame();
66             return;
67         }
68
69         _reachedTargetSpeed = true;
70
71         simulateFlame(false);
72     }
73
74     void simulateFlame(bool isGrowing = true)
75     {
76         if (_rigidbody.velocity.x < 0) return;
77
78         if (isGrowing)
79         {
80             if (flame.transform.localScale.x < 0.25f)
81                 flame.transform.localScale += new Vector3(0.005f, 0.005f, 0.005f);
82         }
83         else
84         {
85             if (_rigidbody.velocity.x > 0 && flame.transform.localScale.x > 0f)
86             {
87                 flame.transform.localScale -= new Vector3(0.005f, 0.005f, 0.005f);
88             }
89         }
90     }
91
92     void OnCollisionEnter(Collision collision)
93     {
94         if (collision.gameObject.CompareTag("Spring"))
95         {
96             _rigidbody.velocity = -_rigidbody.velocity * _springConstant;
97         }
98     }
99
100    void OnApplicationQuit()
101    {
102        _statisticsVelocityX.WriteTimeSeriesToCsv();
103        _statisticsImpulseX.WriteTimeSeriesToCsv();
104        _statisticsPositionX.WriteTimeSeriesToCsv();
105        _statisticsImpulseBothCubesX.WriteTimeSeriesToCsv();
106    }
107 }

```

```

1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using DefaultNamespace;
5  using UnityEngine;
6  using UnityEngine.SceneManagement;
7
8  public class Wurfel2Controller : MonoBehaviour
9  {
10     private Rigidbody _rigidbody;
11     private Statistics _statisticsVelocityX { get; set; }
12     private Statistics _statisticsImpulseX { get; set; }
13     private Statistics _statisticsPositionX { get; set; }
14
15     // Start is called before the first frame update
16     void Start()
17     {
18         _rigidbody = GetComponent<Rigidbody>();
19         _statisticsVelocityX = new(_rigidbody.name, "velocity", "v", "x");
20         _statisticsImpulseX = new(_rigidbody.name, "impulse", "F", "x");
21         _statisticsPositionX = new(_rigidbody.name, "position", "r", "x");
22     }
23
24     // Update is called once per frame
25     void Update()
26     {
27     }
28
29     private void FixedUpdate()
30     {
31         _statisticsVelocityX.AddTimeStep(Time.time, _rigidbody.velocity.x);
32         _statisticsImpulseX.AddTimeStep(Time.time, _rigidbody.mass * _rigidbody.velocity.x);
33         _statisticsPositionX.AddTimeStep(Time.time, _rigidbody.position.x);
34     }
35
36     private void OnCollisionEnter(Collision collision)
37     {
38         if (!collision.gameObject.CompareTag("Wuerfell1")) return;
39         var fixedJoint = gameObject.AddComponent<FixedJoint>();
40         fixedJoint.connectedBody = collision.gameObject.GetComponent<Rigidbody>();
41     }
42
43     void OnApplicationQuit()
44     {
45         _statisticsVelocityX.WriteTimeSeriesToCsv();
46         _statisticsImpulseX.WriteTimeSeriesToCsv();
47         _statisticsPositionX.WriteTimeSeriesToCsv();
48     }
49 }

```