UNIVERSITÄT
KOBLENZ · LANDAU

Fachbereich 4: Informatik

Physikbasierte Hubschraubersimulation basierend auf dem Flugmuster Eurocopter EC 135 mit einem idealisierten Rotormodell

Bachelorarbeit

zur Erlangung des Grades Bachelor of Science (B.Sc.)
im Studiengang Computervisualistik

vorgelegt von

Patrik Schmidt

Erstgutachter:	Prof. Dr.-Ing. Stefan Müller (Institut für Computervisualistik, AG Computergraphik)
Zweitgutachter:	M.Sc. Kevin Keul (Institut für Computervisualistik, AG Computergraphik)

Koblenz, im September 2018

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden.	☐	☐

.....

(Ort, Datum) (Unterschrift)

Inhaltsverzeichnis

Abbildungsverzeichnis

Zusammenfassung

Hubschrauber sind aus heutiger Sicht unverzichtbar. Eine Reihe von Anwendungsgebieten zeigt das Einsatzspektrum, die andere Flugmuster im Vergleich zum Hubschrauber nicht leisten können. Allerdings handelt es sich bei einem Hubschrauber um ein sowohl technologisch als auch physikalisch hochkomplexes System. Entsprechend aufwendig ist die Aus- und Weiterbildung von Piloten. Gerade in den letzten zwei Jahrzehnten hat sich daher die Flugsimulation als wertvolle Ergänzung zum klassischen Training herausgestellt. Mittels Flugsimulatoren ist es möglich, schwierige oder gar gefährliche Situationen bedarfsgerecht nachzuempfinden und zu üben. Im Rahmen dieser Arbeit soll ein vereinfachter Hubschraubersimulator, basierend auf Starkörperkinematik, entwickelt werden. Dabei wird ein idealisiertes Rotormodell angenommen und auf komplexe strömungsmechanische Phänomene verzichtet, um eine Implementation übersichtlich zu illustrieren und echtzeitfähig zu sein. Dabei sind die Module dementsprechend in der Unreal Engine umgesetzt, dass eine Adaption an andere Flugmuster ohne großen Aufwand möglich ist.

Abstract

Helicopters are crucial in today's life. A vast amount of applications prove their range, which are not coverable by other types of aircraft. But they are very complex systems, both, technically and physically. This is one of the reasons why pilot training for helicopters is quite challenging. In the last two decades flight simulators became a supplementary instrument in the educational process of pilots. With flight simulators it is possible to replay uncommon or dangerous situations. In this thesis a simple flight simulator for helicopters will be developed based on rigid body physics. The foundation is a simplified rotor model which omits complex fluid dynamics. This helps to keep the implementation simple and illustrative as well as provide simulation rates at real-time. The modules are implemented within the Unreal Engine in such way, that changing helicopter characteristics is very easy.

1 Einleitung

1.1 Motivation

Im alltäglichen Leben erbringen Hubschrauber eine Vielzahl von Dienstleistungen, die ein weites Kundenportfolio abdecken und damit vielseitige Operationen im täglichen Umgang mit der Hubschrauberfliegerei darstellen. In Ergänzung zur Flächenfliegerei, deren Haupteinsatzgebiet Mittel- und Langstreckenflüge sind, spezialisieren sich die überwiegende Zahl der Hubschraubermuster auf kurze Strecken bis 150km Einsatzradius. Innerhalb dieses Sektors erweisen sich Hubschrauber aufgrund ihrer Bauart als besonders hervorragend für Transportflüge ohne fest angelegten Start/-Landeplatz, für Überwachungsmissionen von Polizeibehörden und ähnlichen oder als Arbeitsgerät im alpinen Bereich. Am präsentesten sind Hubschrauber sicherlich in der seit den 1970er Jahren bestehenden Rettungsfliegerei der Bundeswehr, in Zusammenarbeit mit der ADAC Luftrettung und der Deutschen Luftrettung (DLF). Auf diesem Gebiet stellt der Rettungshubschrauber ein äußerst effizientes und schonendes Rettungsmittel dar, welches gerade fernab von Ballungszentren eine gute notärztliche Versorgung sicherstellt. Doch auch über großen Städten kommen Rettungshubschrauber nicht selten zur Anwendung und fordern vollen Einsatz von Mensch und Maschine bei der Bewältigung komplexer Einsatz- und Gefahrensituationen. Dabei ist der Pilot in ganz besonderem Maße gefordert, das sichere navigieren zum einen und einen sicheren Flugg grundsätzlich zu gewährleisten. Konstruktionsbedingt muss der Pilot dabei einen inhärent instabilen Flugzustand des Hubschraubers abfangen und wie gewünscht korrigieren. Gerade in beengten Situation innerhalb von Städten ist dies überaus anspruchsvoll. Unerlässlich ist es deshalb Piloten gründlich zu schulen und auf (möglichst) alle Unwägbarkeiten und unvorhergesehene Situationen zu trainieren. Immer stärker rückt dabei die Simulatorausbildung, die es den Ausbildern ermöglicht, auch aussergewöhnliche Sachverhalte und Fluglagen gefahrlos zu illustrieren und zu üben, in den Fokus. Die realitätsgetreue Nachbildung des Flugmodells, welches während der Simulation verwendet wird, sollte dabei möglichst hoch sein, die Systemperformance allerdings nicht negativ beeinflussen, sodass eine echtzeitfähige Simulation möglich ist.
Basierend auf den theoretischen Grundlagen der Hubschrauberaerodynamik und den Simulationsmethoden für Festkörper soll im Rahmen dieser Arbeit ein Hubschraubersimulator entwickelt werden. Aufgrund der technischen Komplexität eines Hubschraubers wird dabei ein idealisiertes Rotormodell eingeführt, was eine in Grenzen, realitätsgetreue aber nicht realistische Simulation ermöglicht. Dadurch soll jeder einmal in die Lage versetzt werden, welche verschiedenen Steuereingaben zu welchen Flugzuständen führen.

1.2 Geschichte

1.2.1 Anfänge des Hubschrauberbaus

Luftfahrzeuge faszinieren die Menschen schon seit der Anfangszeit um die Jahrhundertwende als die ersten primitiven Fluggeräte den Kontakt zum Boden verloren. Aber schon viel früher sind der Vorstellungskraft nach einem vogelähnlichen Fluggerät keine Grenzen gesetzt. Erste Erwähnungen, die sich als ein Wunsch nach der Flugfähigkeit, wie sie der Hubschrauber inne hat, reichen mehrere Jahrhunderte vor Christus zurück. Wohl aber am prominentesten ist die zeichnerische Konstruktion von Leonardo Da Vinci aus dem Jahr 1490 [Polte, 2011]. Eindrucksvoll beschreibt Polte in seinem Buch *Hubschrauber - Geschichte, Technik, Einsatz*, wie es Da Vinci vor über 500 Jahren schon gelang, das Grundwesen eines Hubschraubers mit den damaligen Begrifflichkeiten relativ gut zu charakterisieren:

„Die Luft hat Substanz (heute sagen wir Dichte). Wenn man also einen Flügel spiralartig um einen senkrechten Schaft montiert und diesen mit ziemlich hoher Geschwindigkeit dreht, erhebt er sich in die Luft."

Während die Flächenfliegerei relativ bald rasante Fortschritte gemacht hat

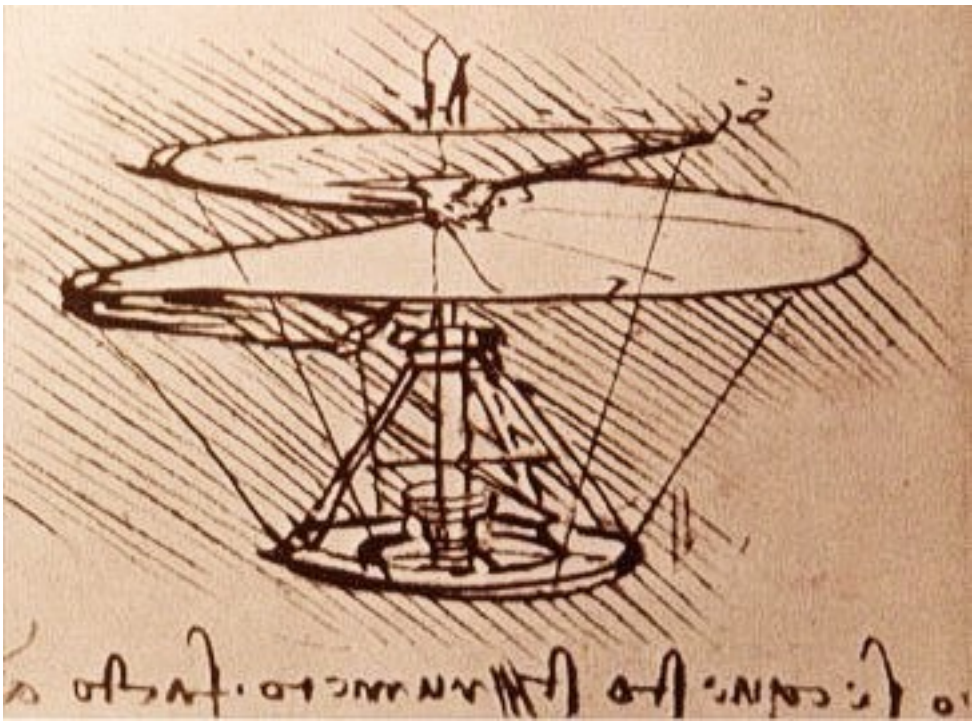

Abbildung 1: Detailaufnahme des Helix Pteron - Der Luftschraube - von Leonardo Da Vinci aus dem Jahr 1490

und sich so schnell etablierte, gab es bei der Entwicklung von Hubschraubern eine ganze Reihe von Hindernissen, die es zu überwinden galt. Ein lange ungelöstes Problem war die Art des Antriebs. Zwar hatte man schon ab 1768 mit den Modellen von J.P. Paucton neue Rotorkonzepte, später folgten auch Koaxialrotoren, allerdings waren diese meist von Muskelkraft angetrieben. Später versuchte man stattdessen Verbrennungsantriebe zu konstrieren, allerdings hatten diese Modelle, basierend auf Holzkohle, Salpeter und Gips, nicht die erforderliche Leistungsausbeute. 1843 nutzte der Engländer Sir Georges Cayley eine Dampfmaschine für den Antriebsstrang, doch auch dies war wenig von Erfolg gekrönt, da das hohe Gewicht der Dampfmaschine die damalige Hubschrauberkonstruktion schlichtweg überforderte. 1907 folgte ein erster erfolgsversprechender Flugversuch der Brüder Louis und Jacques Breguet. Ihr Hubschrauberkonstrukt basierte auf einem 50 PS starkem Antoinette-Kolbenmotor und ließ ihn in 1,5m Höhe schweben. Allerdings musste er von externen Helfern stabilisiert werden. Noch im selben Jahr gelang es Paul Cornu mit seiner Hubschrauberkonstruktion innerhalb von 2 Sekunden in 30cm Höhe frei zu schweben - der erste bemannte Hubschrauberflug. Ab diesem Zeitpunkt wurde die Forschung und Entwicklung von Hubschraubern immer weiter voran getrieben, sicher nicht frei von herben Rückschlägen, so dass auch der junge Pionier Igor Sikorsky nach anfänglichen Versuchen erst 1939 wieder damit anfing Hubschrauber zu entwickeln.

1.2.2 Entwicklung von turbinengetriebenen Hubschraubern

Nach dem Ende des dritten Reiches ist fast die gesamte Hubschraubertechnologie- und Forschung ins Ausland abgewandert. Eine Reihe von führenden Konstrukteuren arbeitet in dieser Zeit für Hersteller wie Bell, Sikorsky oder Aérospatiale. Als 1955 mit der Alouette II der erste turbinengetriebene Hubschrauber entwickelt wurde, war das alltbekannte Problem der fehlenden Antriebsleistung nahezu ad acta gelegt. Ab diesem Zeitpunkt gelang es nach und nach allen Herstellern Muster mit Turbinentriebwerken auf den Markt zu bringen. Basierend auf der BO 105, der Firma Messerschmitt-Bölkow-Blohm, entwickelte seit Ende der 1980er Jahre die Firma Eurocopter (heute Bestandteil der Unternehmensgruppe Airbus Helicopters) aus dem süddeutschen Donauwörth ihren Kassenschlager EC 135. Dieser Hubschrauber ist seit der ersten Erprobung 1994 ein vielseitiges und gefragtes Muster bei Behörden, Rettungsfliegern und privaten Unternehmen. Dabei zeichnet sich die EC 135, siehe Abbildung 2, durch eine Reihe von modernen Konstruktionen aus, die den Piloten entlastet und ihm so ermöglicht, sich auf die eigentlichen Aufgaben zu konzentrieren. Kampa et al. fassen die grundsätzlich projektierten Neuerungen in [Kampa et al., 1997] zusammen. Eines der erwähnten Systeme nennt sich Yaw Stability Augmentation System (kurz Yaw-SAS) und ist als eine Autopilotkomponente integriert.

Abbildung 2: Airbus EC135 [Abarr, 2018]

Das Yaw-SAS dient als so genannter Gierdämpfer und steuert dem Drehmoment, erzeugt durch die Hauptrotorbewegung, entgegen, sodass der Pilot dieses nicht mehr gänzlich selbst ausgleichen muss. Eine auffällige Konstruktionsänderung ist der Heckrotor, welcher bei der EC 135 als Fenestron ausgeführt ist. Durch diese Bauweise erreichen die Ingenieure eine Lärmminderung (zusätzlich zu denen des Hauptrotors), was ihn gerade auch für Einsätze in Lärmschutzbereichen wie Wohngebieten prädestiniert. Eine weitere Neuerung ist der völlige Verzicht auf Schlag- und Schwenkgelenke, sodass das Rotorblatt vom Blattanschluss nahtlos gefertigt ist. In Tabelle 1 und im Typenbuch [Gar, 2012] wird das moderne Leistungssprektrum deutlich, die die EC 135 zu liefern vermag.

1.2.3 Einsatzbereiche

Wie eingangs geschildert gibt es ein breites Spektrum an Einsatzmodalitäten für Hubschrauber. Grundlegend sind dabei Faktoren wie Flughöhe, Reichweite, Zuladung und so weiter. Je nach Blattanzahl und Rotorumdrehungszahl können sich schon hier frappierende Unterschiede in der Eignung für einen entsprechenden Zweck ergeben. So nimmt beispielsweise auch die grundsätzliche Leistungsfähigkeit, da die Luftdichte mit der Höhe abnimmt, des Hubschraubers beim Flug in hohen Lagen ab und der Leistungsbedarf an die Triebwerke steigt. Dies äußert sich beispielsweise in einer gesunkenen Startleistung, was den Piloten dazu veranlassen

Gesamtlänge:	12,16 m
Gesamtbreite:	2,65 m
Gesamthöhe:	3,85 m
Rotordurchmesser:	10,20 m
Höchstgeschwindigkeit:	257 km/h
Reisegeschwindigkeit:	234 km/h
Leergewicht:	1800 kg
Maximale Abflugmasse (MTOW):	2950 kg
Maximale Zuladung:	1110 kg
Dienstgipfelhöhe:	10.000 ft bei Flugmasse > 2835 kg
	12.000 ft bei Flugmasse < 2835 kg
	20.000 ft bei Flugmasse < 2720 kg
Triebwerk:	2 x Turbomeca Arrius 2 B2
Leistung:	je 700 PS Startleistung
	je 570 PS Dauerleistung
Kraftstoffmenge:	593 Liter/ 474 kg im Haupttank
	117 Liter/ 93 kg in den Speisetanks
Kraftstoffverbrauch:	225 l/h
Reichweite:	640 km/ 2,25 Stunden

Tabelle 1: Grundleistungsdaten der EC 135 [Airbus Helicopters, 2014]

kann die Zuladung einzuschränken und besondere Abläufe für diese so genannten High-Altitude-Operations durchzuführen. Unter Berücksichtigung dieser Faktoren haben sich in den letzten Jahrzehnten eine Reihe von Einsatzbereichen und Märkte für Hubschrauber herausgebildet, die in Tabelle 2 dargestellt sind. Fast die Hälfte aller Hubschrauber wird im öffentlichen Sektor eingesetzt, und dort eben auch im Bereich der Notfallrettung (engl. Emergency Medical Service). Ebenfalls lässt sich erkennen, dass sich die Turbine als präferierte Antriebsart mit etwa zwei Drittel auf dem Markt durchgesetzt hat. Kolbenmotoren kommen hauptsächlich in Bereichen zum Einsatz, in denen keine große Performance verlangt ist, Sicherheitsmargen nicht erforderlich sind oder Treibstoffkosten eine hohe Rolle spielen.

1.3 Vor- und Nachteile

Mit zu seinem großen Erfolg haben eine Reihe bautechnisch bedingter Eigenschaften beigetragen. So ist es vielfach möglich, mit einem Hubschrauber auf engstem Raum und ohne jegliche Start- oder Landebahn zu operieren. Damit ist er prädestiniert für Off-Shore Einsätze, die es meist gar nicht erst zulassen, dass entsprechende Fluginfrastruktur bereitgestellt werden kann. Das wohl markanteste Unterscheidungsmerkmal zwischen Hubschrau-

Hersteller	Turbine/n	Kolbenmotor	Gewichtsklassen	
Agusta	1,9 %	-	Leicht, Mono-TW (<6000 lbs)	40 %
Bell	47,9 %	26,6 %		
Enstrom	-	8,9 %	Leicht, Multi-TW (<6000 lbs)	30 %
Eurocopter	28,8 %	-		
Hiller	-	10,3 %	Intermediate (6000-15000 lbs)	25 %
Hughes	-	15,8 %		
MDHC	12,1 %	-	Mittel und schwer (>15000 lbs)	5 %
Robinson	-	25,8 %		
Schweizer	-	4,8 %		
Sikorsky	4,6 %	-		
Sonstige	4,7 %	7,8 %		
Missionen			Antriebsart	
Public Sector: EMS, Polizei, Zoll		45 %	Kolbenmotor	38 %
Geschäftsreise		20 %	Einzelturbine	44 %
Nutztransport		20 %		
Off shore -Einsatz		10 %	Mehrturbinen	18 %
Andere		5 %		

Tabelle 2: Auflistung des zivilen Hubschraubermarktes [Bittner, 2014]

bern und konventionellen Flächenflugzeugen ist der Schwebeflug. Darauf aufbauend ergeben sich vielseitige Einsatzmöglichkeiten. Ebenso ist der reine Vertikalflug eine ausgezeichnete Besonderheit, die dem Hubschrauber eine hohe Wendigkeit verleiht. Doch diese Vorteile kommen alle mit einem hohen Preis an die Leistungsanforderungen, weshalb es auch lange keinen Durchbruch in der Hubschrauberentwicklung gab. Sie bedurfte, wie oben bereits erwähnt, einer enormen Leistungsdichte, sprich hohe Leistung bei geringem Gewicht. Erst moderne Turbinentriebwerke waren in der Lage diese Anforderung zu erfüllen. Daraus ergeben sich weitere Nachteile, die auf die Verwendung von Turbinentriebwerken beruhen. Diese sind kostentechnisch sehr intensiv und haben einen hohen Treibstoffverbrauch. Sicher gibt es dort auch Effizienzsteigerungen über die Jahre, aber grundsätzlich ist ein Hubschrauber mit Turbinentriebwerken teurer im Flug und Unterhalt als ein Hubschrauber oder ein Flugzeug mit Kolbenmotor. Dieser hohe Leistungsbedarf ist eine Grenze für die Reichweite bei Einsätzen mit Hubschraubern. Hier sind Flächenflugzeuge über weite Strecken überlegen. Zudem kommt noch eine konstruktionsbedingte Einschränkung hinsichtlich der maximalen Fluggeschwindigkeit hinzu. Beim Vorwärtsflug werden die Eigengeschwindigkeit und die Rotorblattgeschwindigkeit addiert, beziehungsweise voneinander subtrahiert, im Falle des vorlaufenden oder rücklaufenden Rotorblattes. Dadurch kann es am rücklaufenden Blatt

zu einem verminderten Auftrieb kommen, dem sogenannten Retreating-Blade-Stall. Er ist die Ursache dafür, dass selbst moderne Hubschraubermuster meist nicht über die 300 km/h hinaus kommen. Ebenfalls treten ab diesen Geschwindigkeiten Machzahleffekte auf und man hat mit erhöhten Widerstandskräften, insbesondere an den Blattspitzen, zu rechnen [Abbott and Von Doenhoff, 1959].

2 Grundlagen

2.1 Baugruppen

Dieser Abschnitt soll einen groben Überblick über die Vielzahl an Komponenten eines Hubschraubers geben. Details zu den relevanten Bauteilen sind in den folgenden Unterkapiteln aufgeführt. Diese werden im späteren Abschnitt für die Implementation und die Beschreibung des Flugdynamikmodells wieder aufgegriffen. Eine Übersicht findet sich schematisch in Abbildung 3 wieder.

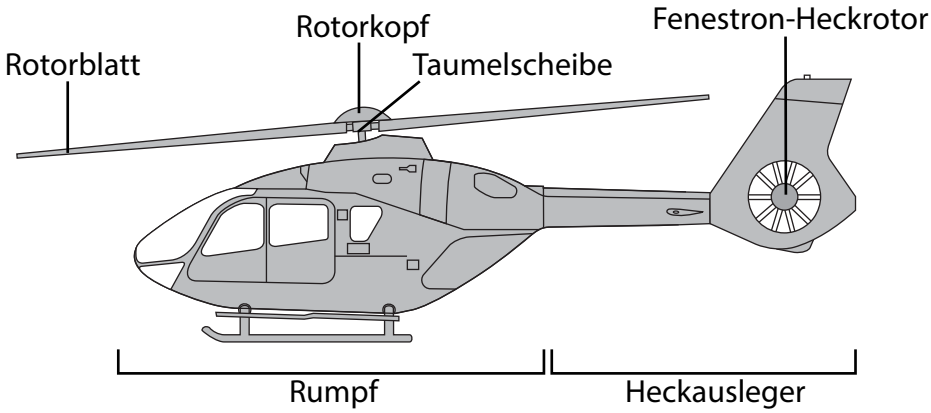

Abbildung 3: Schemaskizze der Baugruppen

2.1.1 Rumpf

Als Rumpf wird im Rahmen dieser Arbeit Folgendes zusammgegefasst bezeichnet:

- Hauptzelle (Cockpit, rückwärtiger Raum)
- Landegestell
- Triebwerke
- Getriebe

Der Rumpf besitzt gemessen am Volumen den größten Beitrag zur Massenträgheit des Hubschraubers. Daher basieren die späteren approximationen dahingehend auch auf den geometrischen Abmessungen des Rumpfes. Auch befindet sich innerhalb des Rumpfes der Großteil der Masse, sowie der Schwerpunkt.

2.1.2 Hauptrotor

Der Hauptrotor befindet sich hinter und oberhalb des Cockpits und ist Bestandteil des Antriebsstrangs der Triebwerke. Der Hauptrotor selbst besteht aus mehreren Rotorblättern, dem Rotormast, der Taumelscheibe und entsprechenden Ansteuerungsgelenken. Die Aufgabe des Hauptrotors ist es (in lokalen Hubschrauberkoordinaten gesehen) vertikalen Schub zu erzeugen, der dafür sorgt, dass die Gewichtskraft des Hubschraubers überwunden wird und dieser abheben kann. Zur gleichen Zeit ist es mittels des Hauptrotors auch möglich, durch zyklische Blattverstellung, Drehmomente zu erzeugen, die dazu führen, dass der Hubschrauber um seine lokalen Achsen rollt beziehungsweise zu giert. Dadurch sind schon einmal zwei der drei Rotationsachsen seitens der Steuerung abgedeckt. Der Bereich der von den Rotorblattspitzen überdeckt wird nennt man Rotorkreis und die Ebene, in der die Rotation abläuft, Rotorebene. Eine wichtige Kenngröße beim Betrieb eines Hubschraubers ist dessen Rotorumdrehungszahl `RPM` (engl. Rounds per minute). Wie aus [Kampa et al., 1997] hervorgeht, behält der Hauptrotor eine konstante Umdrehungsgeschwindigkeit von 395 RPM bei. Über ein intelligentes Triebwerksmanagement wird sichergestellt, dass in verschiedensten Lastsituationen diese Drehzahl nicht schwankt. Die aktuelle Position des Hauptrotors wird über den Drehwinkel ψ definiert. Dieser wird im Allgemeinen als Rotorumlaufwinkel oder Azimut bezeichnet. Bei einem Hauptrotor mit vier Rotorblättern haben alle Rotorblätter einen Abstand von 90° Azimut zueinander.

2.1.3 Heckrotor

Bei allen Single-Mainrotor Hubschraubermustern befindet sich der Heckrotor an einem Ausleger abgesetzt vom Hauptschwerpunkt des Rumpfes. Er ist in einem etwa 90 Grad Winkel relativ zum Hauptrotor vertikal angebracht. Die Hauptaufgabe des Heckrotors besteht darin, das Giermoment um die lokale Hochachse-Achse durch Gegenschuberzeugung auszugleichen. Würde man auf einen Heckrotor gänzlich verzichten, würde ein Hubschrauber anfangen unkontrolliert um die Hochachse zu taumeln und wäre unkontrollierbar. In den meisten Fällen ist die Drehzahl des Heckrotors, wie beim Hauptrotor, konstant und die Anpassung des Schubs wird über eine kollektive Blattansteuerung erreicht. Anders als beim Hauptrotor existiert für den Heckrotor keine zyklische Blattverstellung. Mittels des Heckrotors kann nun auch, zusammen mit dem Hauptrotor, um drei Achsen rotiert werden. Charakteristisch für die EC 135 ist der Fenestron-Heckrotor mit zehn Rotorblättern und einer konstanten Umdrehungsgeschwindigkeit von 3584 RPM. Da die EC 135 mit einem Stabilisierungssystem ausgestattet ist, wird der Heckrotorschub automatisch an das Giermoment des Hauptrotors angepasst, sodass eine stabile Fluglage um die Hochachse ge-

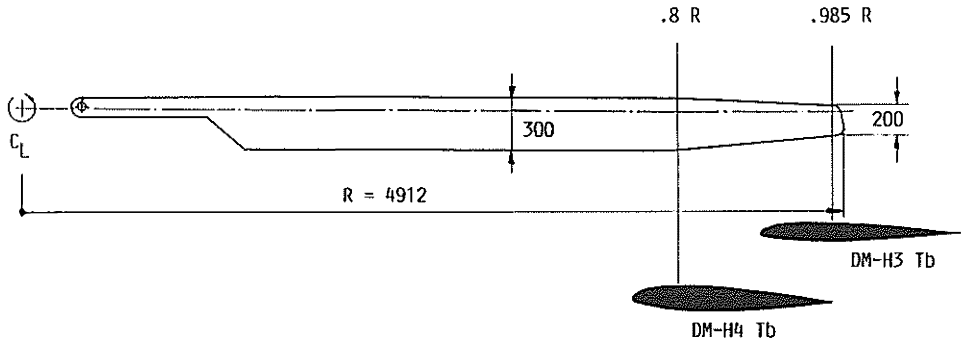

Abbildung 4: Konstruktion des Rotorblattes der EC 135, entnommen aus [G. Polz, 1987]

währleistet werden kann und der Pilot auf diese Art und Weise entlastet wird.

2.1.4 Rotorblatt

Das Rotorblatt ist ein recheckiges Bauteil, welches maßgeblich für die Schuberzeugung verantwortlich ist. Ein Rotorblatt besitzt zur Seite des Rotorkopfes einen Blattanschluss, der entweder ein mechanisches Schlag- und Schwenkgelenk sein kann oder wie bei modernen Mustern durch eine gelenklose Montage erfolgt. Im Falle einer gelenklosen Montage, wie beispielsweise bei der EC 135, nimmt das Rotorblatt alle Torsions- und Schlagbewegungen im Material auf. Daher sind für diesen Typ von Rotorblättern sehr hohe Anforderungen an das eingesetzte Material notwendig. Üblicherweise kommen hier Verbundwerkstoffe aus Glasfaser oder Kohlefaser zum Einsatz. Stellenweise wird auch Titan verwendet. Eine wichtige Eigenschaft eines Rotorblattes ist dessen Blattprofil. Damit beschreibt man den Profilverlauf entlang des Rotorblattes. Mit dem Blattprofil wird die Form, betrachtet im Blattquerschnitt, bezeichnet. Es entscheidet über die aerodynamischen Qualitäten eines Rotorblattes und ist von immenser Bedeutung für die Steuerbarkeit und den so genannten Einsatzenvelope (im Allgemeinen: Leistungsbereich) des Hubschraubers. Man unterscheidet zwischen symmetrischen und unsymmetrischen Rotorprofilen, sowie Sonderformen. Je nachdem welche Profilform bei einem Rotorblatt vorliegt erzeugt es für die anliegenden Einstellwinkel unterschiedliche Auftriebs- und Widerstandskräfte. Eine erste Entwicklung bei der EC 135 hinsichtlich der neuen Rotorblätter gab es schon 1987. Damals noch bei Messerschmitt-Bölkow-Blohm stellten Polz und Schimke in [G. Polz, 1987] ihre bisherige Entwicklung vor. Aus [G. Polz, 1987] ist die in Abbildung 4 dargestellte Konstruktionszeichnung entnommen. Auffällig ist die Verwendung von

zwei Blattprofilen. Bis zu einem Radius von 80 Prozent wird das Profil mit der Bezeichnung *DM-H4* verwendet. An der Blattspitze kommt das Profil *DM-H3* zum Einsatz. Beim Heckrotor findet man ebenfalls das *DM-H3*-Profil. Leider sind diese Blattprofile öffentlich nicht dokumentiert in Bezug auf ihre Leistungsparameter. Deshalb werden für die Simulation Profile mit der Bezeichnung NACA 23012 verwendet. NACA steht für *National Advisory Committee for Aeronautics* und ist 1958 in der NASA aufgegangen [Leishman, 2006]. Das Rotorblattprofil kam, wie in [Kampa et al., 1997] aufgeführt, beim Entwicklungsprototyp BO105/108 zum Einsatz. Weitere Details zu Rotorprofilcharakteristika stehen im Abschnitt 2.2.8.

2.1.5 Taumelscheibe

Um die Steuereingaben des Piloten, die über die Lenkeinrichtungen zum Hauptrotor gehen, vom hubschrauberfesten System in das rotierende Rotorkopfsystem zu übersetzen, verfügt ein jeder Rotor mit kollektiver und zyklischer Blattverstellung über eine Taumelscheibe. Eine solche ist in Abbildung 5 dargestellt. Sie besteht aus einem unteren, festen und oberen, rotierenden Teil. Die Steuergestänge, welche aus dem Cockpit kommen, sind am unteren Teil der Taumelscheibe befestigt. Am oberen Teil führt je ein Gestänge zur Festlegung des Einstellwinkels von der Taumelscheibe zum Rotorblatt. Auf diese Weise ist eine Entkopplung der beiden Bezugssysteme mechanisch realisiert. Je nach dem welchen Drehwinkel der Blattanschluss gerade auf der Oberseite der Taumelscheibe beschreibt, ändert sich dessen Höhe und über ein Steuergelenk an der Rotorblattwurzel der Einstellwinkel des jeweiligen Rotorblattes.

2.2 Aerodynamik

2.2.1 Überblick der aerodynamischen Kräfte

Um die Fluglage und Bewegung eines Flugkörpers (später dann Starrkörper) beschreiben zu können, benötigt man Kenntnis über die einwirkenden Kräfte. Dieser Abschnitt soll vertiefend darauf eingehen, welche Kräfte auftreten und wie sie zu berechnen sind. Grundsätzlich greifen beim Flug eines Flugzeugs oder Hubschraubers vier verschiedene Kräfte an. Wie in Abbildung 6 zu sehen ist, handelt es sich dabei um:

Gewichtskraft Dies ist die Kraft, die der Körper aufgrund der Gravitation erfährt und senkrecht nach unten zum Mittelpunkt des gravitativen Körpers, in unserem Fall der Erde, führt.

Auftriebskraft Die Auftriebskraft ist der Gewichtskraft entgegengesetzt und ist ein Kernelement dessen, warum ein Flugkörper vom Boden abhebt. Sobald die Auftriebskraft der Gewichtskraft überwiegt hebt der Flugkörper ab.

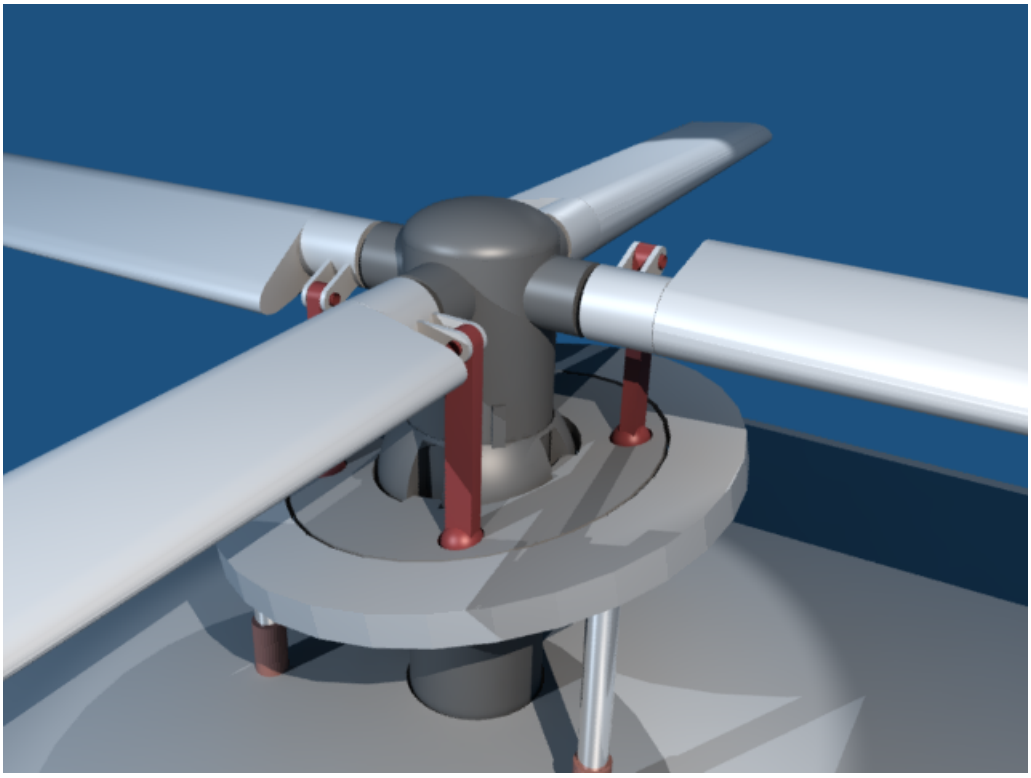

Abbildung 5: Rotor mit zyklisch ausgelenkter Taumelscheibe, Blattsteueranschlüsse (Oberseite), Steuerleitungen (Unterseite)

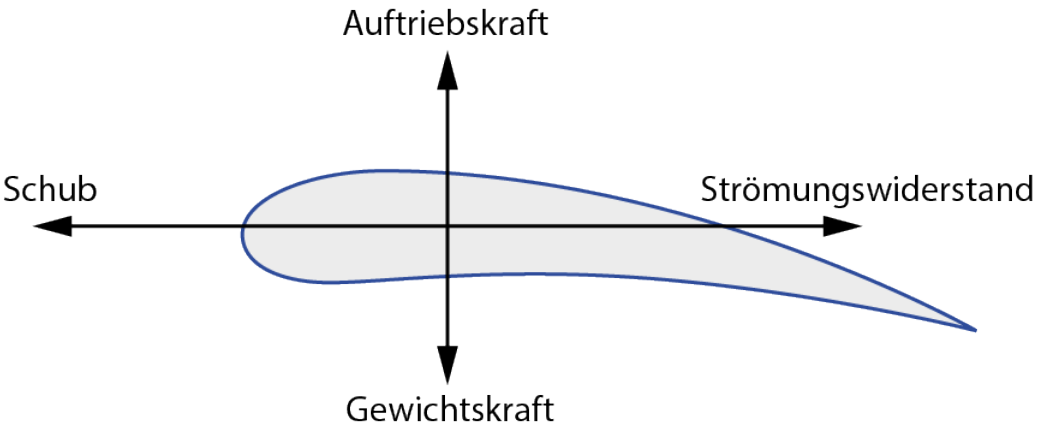

Abbildung 6: Seitenansicht einer Tragfläche mit den entsprechenden aerodynamischen Kräften. Die Anströmung kommt von der linken Seite.

Schubkraft Der Schub richtet sich nach vorne und sorgt für einen Vortrieb. Dadurch wird bei einem Flächenflugzeug die Anströmung der Tragflächen erreicht. Bei Hubschraubern unterscheidet man zwischen Schub als Vorwärtsgeschwindigkeit oder der Begriff wird synonym für den Auftrieb verwendet.

Widerstandkraft Dabei handelt es sich um eine Gegenkraft zur Vorwärtsbewegung und ist dieser entgegengesetzt. Es gibt eine Reihe von Widerstandskräften, mehr oder weniger einfach zu ermitteln. Der induzierte Luftwiderstand, der durch die Ablenkung des Luftstroms am Rotorblatt erzeugt wird, ist einer der Hauptbestandteile. Er wird allerdings betragsmäßig bei zunehmender Fluggeschwindigkeit kleiner und fällt nicht mehr groß ins Gewicht. Dahingehend wächst mit steigender Fluggeschwindigkeit der sogenannte Schubspannungswiderstand. Vereinfacht lässt er sich mit der Reibung der am bewegten Körper haftenden Fluidteilchen (hier Luft) mit den vorbeiströmenden Teilchen erklären. Der Schubspannungswiderstand ist dabei von der betroffenen Fläche abhängig. Eine sehr detaillierte Betrachtung der aerodynamischen Zusammenhänge in Bezug auf die Widerstandskräfte von Profilen bietet [Abbott and Von Doenhoff, 1959].

2.2.2 Gewichtskraft

In diesem Fall gehen wir von dem Begriff der trägen Masse m_t aus, wie sie Anwendung in den Newton'schen Gesetzen findet. In Abhängigkeit von der Masse unseres Hubschraubers (hier genauer TOW, engl. Take Off Weight) berechnet sich die Gewichtskraft über $F = m_t \cdot a$, wobei a hier der Erdbeschleunigung mit $g \approx 9,81 \frac{m}{s}$ entspricht .

2.2.3 Auftrieb

Der Begriff Auftrieb lässt sich in statischer und dynamischer Auftrieb unterscheiden. Statischer Auftrieb entsteht dabei aus dem hydrostatischen Druck wie bei Heliumballons oder der Wasserverdrängung bei Schiffen. Er sei nur der Vollständigkeit halber erwähnt. Für die Simulation von Interesse ist der dynamische Auftrieb. Wird ein Körper von einem Fluid umströmt, in diesem Fall die Tragfläche von Luft, wirkt eine Auftriebskraft senkrecht zur Richtung aus der die Anströmung stammt. Diese Kraft kommt dadurch zu Stande, dass die Luftströmung an der Unterseite nach unten abgelenkt wird, sie wird folglich nach unten beschleunigt. Folgernd aus dem Newton'schen Axiomen ergibt sich dem entsprechend eine Gegenkraft - die Auftriebskraft nach oben. Der Coanda-Effekt sorgt dafür, dass die laminare Luftströmung der Oberflächenkontur folgt. Die geschieht aber nur bis zu einem gewissen kritischen Winkel (dem so genannten Stall-Winkel),

der beim Überschreiten zur Folge hat, dass sich die Luftströmung von der Flügelkontur ablöst. Dieses Ablösen macht sich in einer schlagartig verminderten Auftreibskraft bemerkbar und ist in jedem Fall ein kritischer und zu vermeidender Flugzustand.

2.2.4 Schubkraft

Wie bereits eingangs erwähnt kann der Schub mehrdeutig interpretiert werden. Bei Flächenflugzeugen ist es die durch den Propeller oder die Triebwerke erzeugte Vorwärtsgeschwindigkeit entlang der Rollachse des Flugzeugs. Bei Hubschraubern findet sich ein abgewandelter Sprachgebrauch. Hier wird mit Schub die Aufwärtsbewegung entlang der Hochachse bezeichnet wie sie bei Eingabe von kollektivem Pitch entsteht. Allerdings kann man mit Schub auch die horizontale Geschwindigkeitskomponente beschreiben, die einen Hubschrauber zur Bewegung in Richtung der Z-Achse nach oben veranlasst.

2.2.5 Luftwiderstand

Der Luftwiderstand beziehungsweise der Strömungswiderstand wirkt der Bewegung des Körpers durch die Luft entgegen. Examplarisch ist er zu beobachten, wenn man aus einem fahrenden Auto die Hand aus dem Fenster streckt und die Handflächen senkrecht zur Fahrtrichtung aufstellt. Unmittelbar vergrößert sich der Luftwiderstand und sorgt dafür, dass die Hand nach hinten gedrückt wird. Ähnliches geschieht mit jedem Körper bei der umströmten Bewegung. Es gilt in der Simulation sowohl für den Rumpf als auch für die aerodyamisch wirksamen Flächen wie Rotorblätter oder Stabilisatoren. Abhängig ist die Strömungswiderstandskraft von der Fluiddichte ρ, der Viskosität des Fluids η, der charakteristischen Länge L des Flügeprofils und der Anströmungsgeschwindigkeit v. Durch die Rotation des Hauptrotors und der damit erzeugten Widerstandskraft wird eine Gegenkraft erzeugt, die dazu führt, dass ein Drehmoment um die Hochachse des Hubschraubers wirkt. Die Folge: Der Hubschrauber giert um die Hochachse und muss stabilisiert werden. Wie diese Stabilisierung unter zuhilfenahme des Heckrotors geschieht, wurde bereits in Abschnitt 2.1.3 erläutert. Weitere Details zu den Berechnungen der Drehmomente folgen in Abschnitt 3.4.3.

2.2.6 Auftriebsbeiwert

Der Auftriebsbeiwert ist eine dimensionslose Kenngröße in der Aerodynamik und ist ein Maß für den dynamischen Auftrieb. Ausgehend von den Darstellungen in [Surek and Stempin, 2007] wird er hier wie im Englischen

mit c_l für Lift bezeichnet und es gilt:

$$c_l = \frac{F_l}{q \cdot A} \tag{2.1}$$

mit

$$q = \rho \cdot v^2 \tag{2.2}$$

In Gleichung (2.1) wird der Auftriebsbeiwert aus der Auftriebskraft F_l, dem Staudruck q und dem Flächeninhalt A der angeströmten Fläche ermittelt. ρ bezeichnet die Luftdichte und ist abhändig von der Höhe und Temperatur. Ausgehend von Gleichung (2.1) erhalten wir nach dem Einsetzen des Staudrucks die Auftriebskraft wie in Gleichung (2.3).

$$F_l = c_l \cdot \frac{\rho}{2} \cdot v^2 \cdot A \tag{2.3}$$

2.2.7 Widerstandsbeiwert

Analog zum Auftriebsbeiwert, gibt es den Widerstandsbeiwert, der gerade aus dem Automobilbau auch als c_W-Wert bekannt ist. Der Widerstandsbeiwert c_d (engl. drag) ist ähnlich wie Gleichung (2.4) definiert.

$$c_d = \frac{F_d}{q \cdot A} \tag{2.4}$$

Ebenfalls können daraus, siehe Gleichung (2.5), die Widerstandskräfte abgeleitet werden, die sich im Vergleich zu (2.3) nur im Beiwert unterscheidet.

$$F_d = c_d \cdot \frac{\rho}{2} \cdot v^2 \cdot A \tag{2.5}$$

2.2.8 Polardiagramme

Mit Hilfe der Polardiagramme (oder Lilienthalpolare) werden die Auftriebskoeffizienten und Widerstandskoeffizienten aufgetragen. Für den Auftriebsbeiwert ist in Abbildung 7 einmal beispielhaft aufgeführt. Sie sind charakteristisch für ein bestimmtes Rotorprofil und zeigen die Abhängigkeit der entsprechenden Koeffizienten vom jeweiligen Anströmungswinkel. Schön ist auch der kritische Anstellwinkel zu erkennen, ab dem die laminare Strömung in eine turbulente Strömung über geht, bis es zur Ablösung derer von der Tragfläche kommt. Schlussendlich resultiert dies in einem Strömungsabriss und einem radikal verminderten Auftrieb. Auf der X-Achse sind die Verschiedenen Anstellwinkel aufgetragen. Geht man von diesen Werten senkrecht nach oben und projeziert den Schnittpunkt mit dem Graphen auf die Y-Achse erhält man den entsprechenden Auftriebsbeiwert. Die Daten, welche in den Polardiagrammen und dem Hubschrauberent-

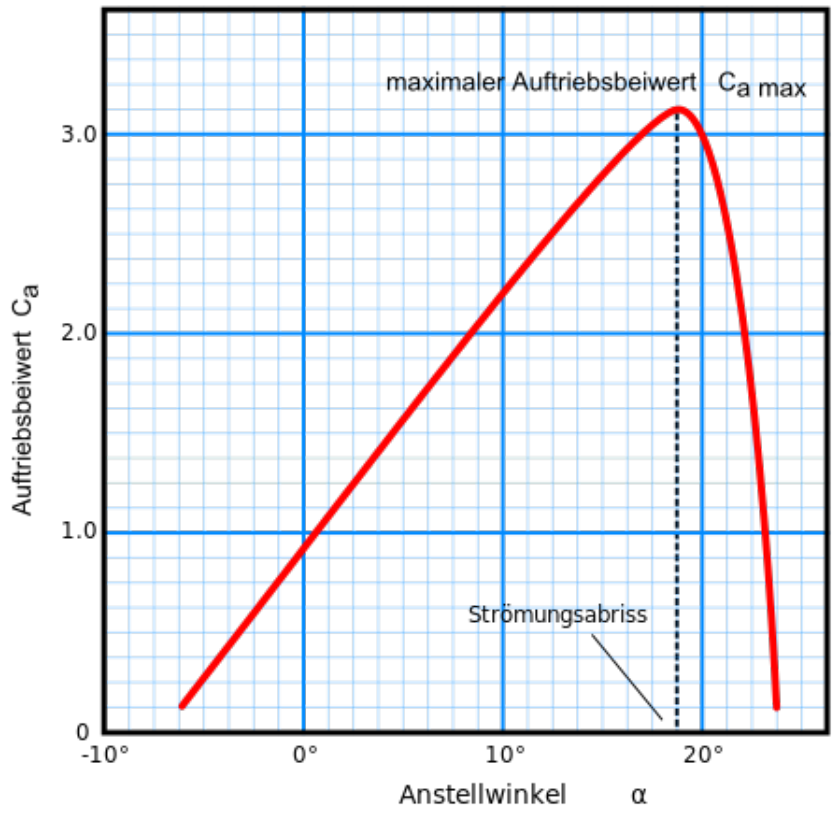

Abbildung 7: Visualisierung des Auftriebsbeiwertes C_a (engl. C_l) mittels eines so genannten Polardiagramms. [Grahamuk, 2012]

wurf Anwendung finden, sind meist experimentellem Ursprung und stammen aus Versuchen in Windkanalanlagen. Auf der Webseite Airfoiltools[1] findet sich eine Datenbank mit einer Vielzahl an Rotorprofilen, sowohl für Flugzeuge, als auch für Hubschrauber. Eine weitere ergiebige Quelle sind die Webseiten[2] der UIUI Applied Aerodynamics Group der University of Illinois. Dort finden sich auch viele Daten, gerade von europäischen Flugzeugherstellern oder Forschungseinrichtungen, wie beispielsweise dem Institut für Aerodynamik und Strömungsmechanik[3]. Insbesondere die Ergebnisse der Abteilung Hubschrauber[4] (AS-HEL), welche in der Vergangenheit auch schon mit Eurocopter kooperierten. Leider sind gerade die

[1] http://airfoiltools.com/
[2] https://m-selig.ae.illinois.edu/ads.html
[3] https://www.dlr.de/as
[4] https://www.dlr.de/as/desktopdefault.aspx/tabid-192/402_read-571/

Daten nicht öffentlich verfügbar, die sich in kommerzieller Verwertung, sprich dem Produktiveinsatz bei Hubschraubern befinden. Darunter fallen auch die genaueren Profildaten der Rotorblätter, wie sie bei der EC 135 zum Einsatz kommen.

2.3 Hubschraubersteuerung

2.3.1 Kollektive Blattansteuerung

Wenn der Pilot den seitlichen Hebel zur kollektiven Blattverstellung betätigt, wie in Abbildung 8 erkennbar, werden alle Rotorblätter gleichzeitig und im gleichen Maße im Einstellwinkel geändert. Diese Änderung wird dadurch erreicht, dass die Taumelscheibe parallel zur Rotormastachse hoch oder runter verschoben wird. Dadurch ändern sich die übertragenen Wege der Steuergestänge von der oberen Hälfte der Taumelscheibe gleichmäßig in allen Quadranten. Mittels der kollektiven Blattverstellung wird ein Schub senkrecht zur Rotorkreisscheibe erzeugt.

2.3.2 Zyklische Blattansteuerung

Um einen Hubschrauber rollen oder nicken zu lassen, sind Drehmomente um die entsprechenden Achsen notwendig. Dazu dient die zyklische Blattansteuerung, die zusätzlich zum kollektiven Pitch mit eingemischt wird. Dieses Mischen der Steuerinputs geschieht wieder durch die Taumelscheibe. Wird der Stick vom Piloten nach vorne bewegt, so kippt die Taumelscheibe um den gleichen Betrag nach vorne. Dadurch werden nur die Gestängepositionen der Rotorblattanschlüsse am stärksten beeinflusst, die sich auf null Grad, beziehungsweise 180 Grad auf der Rotorkreisscheibe befinden. Dazwischen wird linear interpoliert.

2.3.3 Heckrotoransteuerung

Die Heckrotoransteuerung geschieht mittels der beiden Fußpedale für den Piloten. Mit Hilfe derer kann er den kollektiven Blattanstellwinkel der Heckrotosrblätter verstellen. Dadurch entsteht ein Drehmoment um die Hochachse des Hubschraubers, welches den Hubschrauber entweder nach links oder rechts gieren lässt.

2.3.4 Gesamtsystem

Die wie in [Wall, 2015] aufgeführte Formel (2.6) fasst die kollektive und zyklische Blattverstellung wie folgt zusammen:

$$\theta = \theta_0 + \theta_C \cdot \cos\psi + \theta_S \cdot \sin\psi \tag{2.6}$$

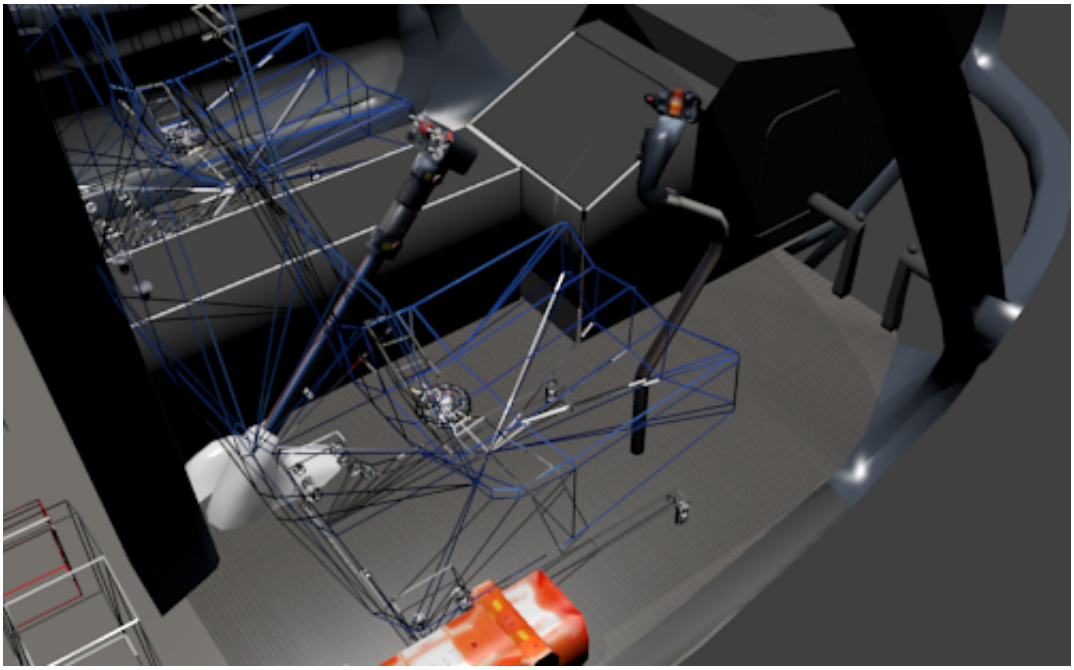

Abbildung 8: Cockpit-Ansicht mit Steuereinrichtungen. Der Pilotensitz ist als Wireframe gerendert. Links davon befindet sich der Hebel für die kollektive Blattverstellung. Vor dem Sitz befindet sich der Flight-Stick für die zyklische Blattverstellung. Am Fussende des Sitzplatzes sind zwei Fußpedale für den Yaw-Input zu erkennen.

Mit θ_0 bezeichnet man die kollektive Blattverstellung, diese wird einfach als Term für alle Rotorblätter übernommen. Die zyklische Blattverstellung hat dahingehend periodischen Einfluss auf die Blattanstellwinkel und ist daher von der Blattposition im Umlauf abhängig und daher von ψ. Um dem Rechnung zu tragen sind die Quersteuerwinkel θ_C und Längssteuerwinkel θ_S jeweils mit dem Sinus, beziehungsweise dem Kosinus multipliziert und ergeben dadurch eben die richtige Einmischung des Gesamteinstellwinkels.

2.4 Impulstheorie

Die Impulstheorie sei hier nur der Vollständigkeit halber erwähnt, da sie im Grunde lediglich grobe Abschätzungen über die Flugleistung eines Hubschraubers zulässt. Für den Schwebeflug liefert sie aber ganz brauchbare Ergebnisse. Es werden beispielsweise die Anzahl der Rotorblätter und deren Rotorprofilcharakteristiken ausser acht gelassen. Dennoch verwendet die Blattelementtheorie im folgenden Abschnitt diese Näherungen, um Gleichungen zu vereinfachen und Parameter zu schätzen.
Ursprünglich wurde sie von Rankine [Rankine, 1865] und Froude [Froude, 1878]

entwickelt und später von Betz (1920) erweitert . Dabei bezog sich die Forschung aber zuerst auf Schiffspropeller. Ausgangslage der Impulstheorie, wie sie von Johnson in [Johnson, 1994] beschrieben wird, ist die Annahme einer so genannten Aktuatorscheibe, die symbolhaft für die (Haupt-)Rotorebene stehen soll. Im Querschnitt gesehen befinden sich oberhalb dieser Aktuatorscheibe die Ebene 0, direkt über der Scheibe die Ebene 1, direkt darunter die Ebene 2 und in weiter Distanz die Ebene ∞. Ein Schema dessen ist in Abbildung 9 dargestellt. Folgende Idealisierungen liegen der Impulstheorie nach [Johnson, 1994], [Wall, 2015] und [Bittner, 2014] zugrunde:

- Die Aktuatorscheibe wird als unendlich dünn angesehen und besitzt keinen Luftwiderstand.
- Sowohl Schub als auch die Luftgeschwindigkeiten sind homogen über die Aktuatorscheibe verteilt. Insbesonders trifft dies auch auf den Wurzelbereich des Rotorblattes zu, welcher unprofiliert ist.
- Die Anzahl und Form der Rotorblätter werden nicht berücksichtigt.
- Ebenso wenig findet die Rotorbewegung Anwendung.
- Strömungen werden als reibungsfrei und inkompressibel betrachtet.

2.4.1 Erhaltungssätze

Grundlage der Impulstheorie sind die Erhaltungssätze. Demnach gilt dem Massenerhaltungssatz folgend, dass der einströmende Massenstrom durch ein Volumen gleich dem ausströmenden Massenstrom ist. Dargestellt ist dies in Gleichung 2.7.

$$\iint_S \rho\vec{V} \cdot d\vec{S} = 0 \tag{2.7}$$

S steht dabei für die Oberfläche mit der der Massenstrom $\rho\vec{V}$ einströmt. Der Impulserhaltungssatz, wie in Gleichung 2.8, besagt, dass eine Kraft, erzeugt durch eine Strömung, gleich der Impulsänderung pro Fläche ist. Die angegebene Kraft wird vom Rotor erzeugt und gemäß dem dritten Newton'schen Gesetz, übt die Strömung demzufolge eine gleich große Gegenkraft auf den Rotor aus.

$$\vec{F} = \iint_S pd\vec{S} + \iint_S (p\vec{V} \cdot d\vec{S})\vec{V} \tag{2.8}$$

Letztlich fehlt noch der Energieerhaltungssatz definiert wie in Gleichung 2.9 dargestellt.

$$W = \iint_S \frac{1}{2}(\rho\vec{V} \cdot d\vec{S})|\vec{V}|^2 \tag{2.9}$$

Sie besagt nichts anderes, als dass die Arbeit W in kinetische Energie, welche die Strömung ab dann inne hat, umgewandelt wird.

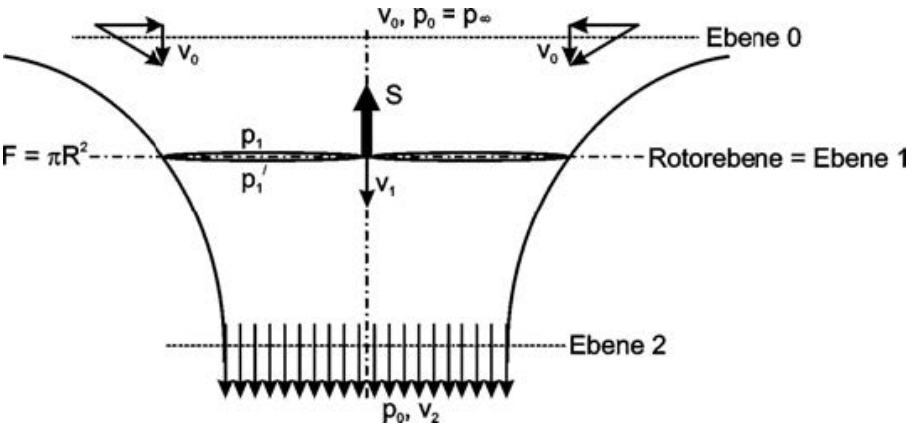

Abbildung 9: Störmungsverlauf mit Aktuatorscheibe [Bittner, 2014]

2.4.2 Schwebeflug

Im Schwebeflug beträgt die in Abbildung 9 gekennzeichnete Geschwindigkeit v_0, oberhalb der Ebene 0, gleich null. Da in der Impulstheorie eine inkompressible, eindimensionale Strömung [Wall, 2015] angenommen wird, lässt sich Gleichung 2.10 dahingehend umformen, dass die Definition des Massenstroms $\dot{m}$ wie folgt lautet:

$$\iint_n \rho p \vec{V} \cdot d\vec{S} = \dot{m} = \rho A v_i = \rho A_n v_{in} \qquad n = 1, 2, \infty \tag{2.10}$$

Davon ausgehend, dass der Strömungszylinder seitlich nicht durchströmt wird, kann jede Fläche A als Querschnittsfläche separat betrachtet werden. Die Gleichung 2.11 gibt den Schub an, der durch einen Impulszuwachs zustande kommt. Da, wie oben angemerkt, die induzierte Geschwindigkeit v_0 oberhalb der Rotorebene von der Aktuatorscheibe unbeeinflusst bleibt, ist der zweite Summand in Gleichung 2.11 gleich Null.

$$T = \iint_\infty \rho(\vec{V} \cdot d\vec{S})\vec{V} - \iint_0 \rho(\vec{V} \cdot d\vec{S})\vec{V} = \iint_\infty \rho(\vec{V} \cdot d\vec{S})\vec{V} = \dot{m} v_{i\infty} \tag{2.11}$$

Eine detaillierte Herleitung und Ergänzungen sind [Wall, 2015] zu entnehmen. Ebenso für die Betrachtungen im Steigflug. Induzierte Geschwindigkeit beschreibt in diesem Kontext die lokale Luftgeschwindigkeit des Massenstroms innerhalb der Strömung. Typische Werte für die induzierte Geschwindigkeit beim Schwebeflug in Ebene 2 sind etwa 6 $\frac{m}{s}$.

2.5 Blattelementtheorie

Auf Grundlage der Blattelementtheorie, wie bei [Bittner, 2014] und [Wall, 2015], findet die Simulation der aerodynamischen Kräfte in der Simulation statt.

Dabei wird das Rotorblatt, wie in Abbildung 10, entlang der Y-Achse in Profilabschnitte unterteilt und anhand derer werden die lokalen Auftriebs- und Luftwiderstandskräfte ermittelt. Ergänzend zur Impulstheorie werden bei der Blattelementtheorie Form und Geschwindigkeit der Rotorblätter mit berücksichtigt. Für ein Blattelement können mit den folgenden Formeln die Auftriebskraft L und die Widerstandskraft D je Blattelement berechnet werden:

$$L = \frac{1}{2} C_l \cdot \rho \cdot l \cdot b \cdot v^2 \tag{2.12}$$

$$D = \frac{1}{2} C_d \cdot \rho \cdot l \cdot b \cdot v^2 \tag{2.13}$$

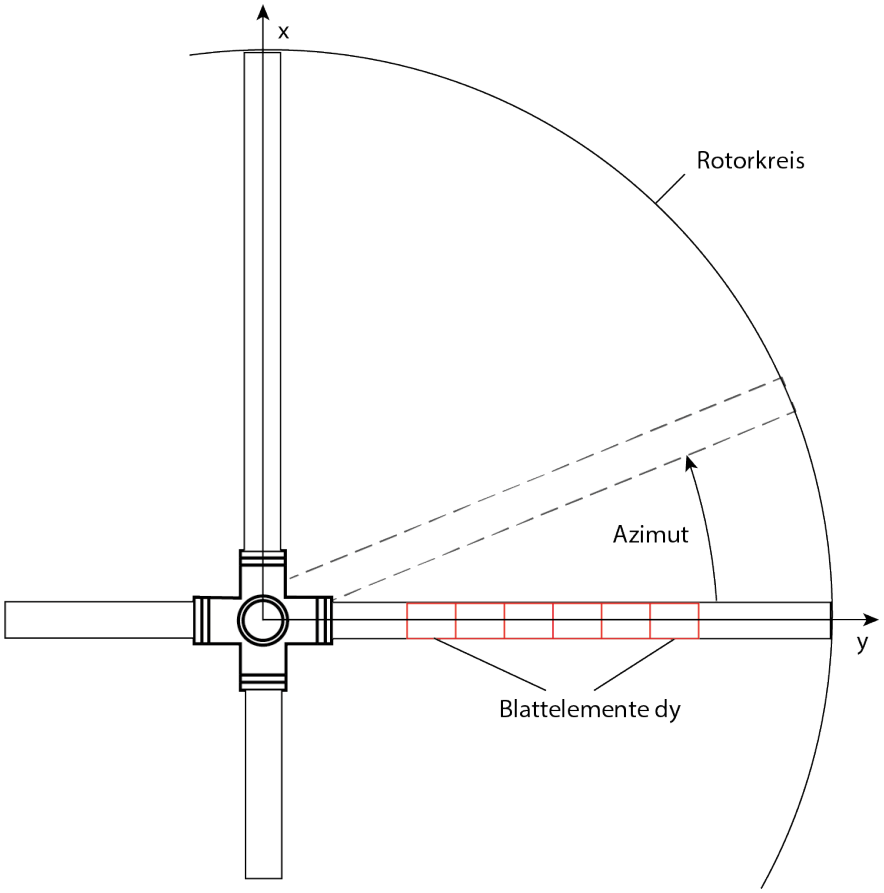

Abbildung 10: Aufsicht der Rotorebene mit angedeuteten Blattelementen dy (rot)

2.5.1 Lokale Anströmung

Die in 2.12 und 2.13 angegebene Anstömungsgeschwindigkeit v ist über die Blattlänge nicht gleichverteilt. Wie in Gleichung 2.14 zu sehen ist hängt die-

se zum einen vom Radius Ωy ab, dem Schwenkwinkel ζ und dem Torsions- und Steuerwinkel θ. Da es sich bei der EC 135, wie in Abschnitt 2.1.4 um einen Hubschrauber mit Rotor ohne Schlag- und Schwenkgelenk handelt, kann der Winkel θ vernachlässigt werden. Dies stellt eine Optimierung für die Simulation dar, führt aber zwangsläufig zu numerischen Ungenauigkeiten.

$$V_T = \Omega y + V_\infty \cos\alpha \sin\psi - V_\zeta(y,\psi) + V_\theta \sin\theta \tag{2.14}$$

Schlussendlich führt uns das zur Berechnung des Gesamtschubs für den Rotor:

$$V_T = N_b \int_0^R L' dy = N_b \int_0^R \frac{\rho}{2} c\Omega^2 y^2 C_l dy \tag{2.15}$$

Die Rotorblattanzahl N_b findet hier, entgegen der Impulstheorie, berücksichtigung. Ebenso fließen Rotorprofilcharakteristika in Form des Auftriebbeiwertes C_l mit ein. Die restlichen Parameter sind lediglich die Luftdichte ρ, die Rotordrehfrequenz Ω, sowie die Anströmungsfläche des lokalen Blattelementes.

3 Rigid Body

3.1 Grundlagen

Unter dem Begriff Rigid Body versteht man in der Physik, insbesonders in der klassischen Mechanik, einen Starrkörper. Dieser besteht aus einer Anzahl von Massepunkten, die in einer festen Verbindung zueinander stehen und (im Falle von inkompressiblen Körpern) einen konstanten Abstand zueinander besitzen. Anders als bei Punktmassen ist die physikalische Simulation von Rigid Bodies um einiges umfangreicher, da bei ihnen die Zustandbeschreibung durch die bloße Angabe der Position nicht mehr ausreichend ist. Zusätzlich zu einer Position $\vec{x} = (xyz)^T$ gibt es jetzt im, im dreidimensionalen, auch noch eine Orientierung $q = (wxyz)^T$. In diesem Fall dargestellt als Quaternion. Die Berechnung der Position wird im Abschnitt 3.3 erläutert. Die Gesetzmäßigkeiten hinsichtlich der Rotationen sind unter 3.4 zusammengefasst erklärt. Es sei an dieser Stelle darauf hingewiesen, dass sowohl die translatorische Bewegung als auch die Rotation getrennt betrachtet werden kann und zum Schluss eines Simulationsdurchlaufs zusammen gerechnet werden. Eberly hat in [Eberly, 2015] einen State-Vektor zur Grundlage seines Starrkörpers gemacht. Dieser State-Vektor ist durch

$$S(t) = \begin{bmatrix} \vec{x}(t) \\ q(t) \\ \vec{p}(t) \\ \vec{L}(t) \end{bmatrix} \tag{3.1}$$

gegeben und definiert den Zustand eines einzelnen Starrkörpers zum Zeitpunkt t. $\vec{p}(t)$ beschreibt dabei den linearen Impuls und $\vec{L}(t)$ den Drehimpuls. Für den linearen Impuls gilt $\frac{\Delta\vec{p}}{\Delta t} = \vec{F}$, beziehungsweise $\vec{p} = m\vec{v}$, was sich aus den Newton'schen Axiomen ableiten lässt.

3.2 Newton'sche Axiome

Die kommenden Abschnitte definieren die Newton'schen Axiome und geben direkt im Anschluss ein kurzes Beispiel der Anwendung innerhalb der Simulation an. Die drei Axiome sind von Tipler wie folgt definiert [Tipler et al., 2015].

3.2.1 Trägheitsprinzip

Definition 3.1: Trägheitsprinzip

Ein Körper bleibt in Ruhe oder bewegt sich geradlinig mit konstanter Geschwindigkeit weiter, wenn keine resultierende äußere Kraft auf ihn wirkt.

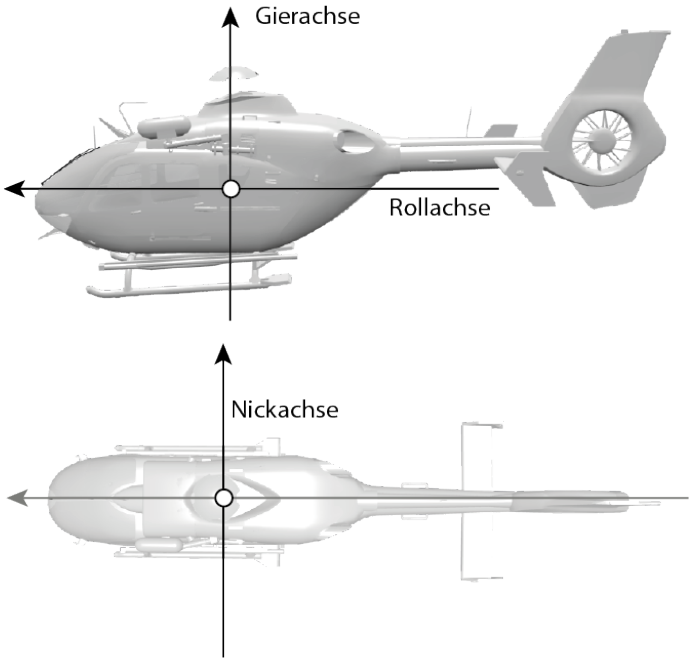

Abbildung 11: Darstellung des lokalen Koordinatensystems. Mittig der manuell festgelegte Massenschwerpunkt.

Von Bedeutung ist dieses Gesetz für die Simulation in der Gestalt, dass wir zur Zustandsänderung des Hubschraubers, in dem Fall der Position und Orientiertung, Kräfte aufbringen müssen. Kräftefrei ist der Hubschrauber nicht, da während der Simulation eine Reihe von verschiedenen Kräften auf die Rotorblätter und den Rumpf einwirkt. Allerdings werden zu Gunsten der Interaktivität eine Vielzahl von Kräften vernachlässigt oder approximiert, etwa Reibung und Dergleichen.

3.2.2 Aktionsprinzip

Definition 3.2: Aktionsprinzip

Die Beschleunigung eines Körpers ist direkt proportional zu der auf ihn wirkenden Gesamtkraft, wobei die Proportionalitätskonstante der Kehrwert der Masse ist. Somit gilt:

$$a = \frac{F}{m} \quad mit \quad F = \sum F_i.$$

Diese Gesetzmäßigkeit bildet einen zentralen Punkt der Simulation. Es werden alle auftretenden Kräfte F aufsummiert. Anschließend wird dann mit der Masse des Hubschraubers m dessen Beschleunigung a berechnet. Dies geschieht durch Integration mit Hilfe des Semi-Impliziten Euler-Verfahrens (siehe Abschnitt 4.2.2). Im linken Teil der Gleichung steckt allerdings noch eine weitere wichtige Eigenschaft unseres Körpers, in Form der Masse m. Die Masse eines Körpers ist im übertragenen Sinne die Eigenschaft, wie hoch der Widerstand gegen Zustandsänderungen durch äußere Kräfte ist. Sinnvollerweise spricht man in diesem Fall auch von träger Masse. Je höher die Masse ist, desto größer muss die einwirkende Kraft sein, um eine gleich große Beschleunigung zu erreichen. Anschaulich erklärt: Ein voll beladener Hubschrauber muss mehr Auftriebskraft (und damit mehr Leistung) aufbringen, um abzuheben, als ein leerer Hubschrauber.

3.2.3 Wechselwirkungsprinzip

Definition 3.3: Wechselwirkungsprinzip

Wenn zwei Körper miteinander wechselwirken, ist die Kraft $F_A^{(B)}$, die der Körper B auf den Körper A ausübt, gleich groß, aber entgegengesetzt gerichtet der Kraft $F_B^{(A)}$, die der Körper A auf den Körper B ausübt. Somit gilt:

$$F_A^{(B)} = -F_B^{(A)}.$$

Häufig wird dieses Prinzip mit dem Begriff *Actio = Reactio* übersetzt und beschreibt das Verhalten von zwei oder mehr Objekten zueinander. Das einfachste Beispiel dafür ist eine Rakete. Während des Fluges stößt sie Masse in Form von Brandgasen aus ($F_A^{(B)}$) und wird mit gleich großer Kraft ($-F_B^{(A)}$) entgegengesetzt beschleunigt. Angewendet auf den Hubschrauber begegnet man dem Wechselwirkungsprinzip an vielen Stellen. So erzeugt die Drehbewegung des Hauptrotors eine Kraftübertragung über die Wellen auf die Rotorblätter und umgekehrt wirkt eine Gegenkraft, die den Hubschrauber veranlasst um die Hochachse zu gieren. Oder aber man betrachtet den stationären Flug (Schwebe). Dort erzeugt der durch den Hauptrotor nach unten gerichtete Luftstrom einen Auftrieb, der erforderlich, ist um die Masse des Hubschraubers in der Schwebe zu halten.

3.3 Kinematik

Zur Berechnung der Position wird der Massenschwerpunkt des Starrkörpers berechnet. Alle Massepunkte sind definiert mit einer Masse und dem zugehörigen Ortsvektor $\vec{r}$ vom Masseschwerpunkt. Grundsätzlich wird die

Translation des Starrkörpers für den Masseschwerpunkt berechnet, alle anderen Massepunkte können dann über den jeweiligen Ortsvektor entsprechend aktualisiert werden.

3.3.1 Geschwindigkeit

Wie eingangs bereits erwähnt, werden Translation und Rotation getrennt voneinander betrachtet. In diesem Abschnitt geht es demnach um die lineare Geschwindigkeit, die der Hubschrauber im dreidimensionalen Raum hat. Der Hubschrauber besitzt eine Position in der 3D-Welt mit $\vec{x} = (xyz)^T$, bezeichnet als Ortsvektor. Darauf aufbauend ist die Geschwindigkeit als

$$\vec{v} = \frac{\Delta r}{\Delta t} \tag{3.2}$$

definiert und hat die Einheit $\frac{m}{s}$. Mit anderen Worten, die Änderung der Position pro Zeitabschnitt. Die Ermittlung der Geschwindigkeit ist der erste Baustein für die Simulation der translatorischen Bewegung. Eine analoge Beschreibung für die Rotation folgt in Abschnitt 3.4.2.

3.3.2 Beschleunigung

Die Beschleunigung ist die zeitliche Ableitung der Geschwindigkeit $\vec{v}$ also

$$\vec{a} = \frac{\Delta v}{\Delta t} \quad . \tag{3.3}$$

Sie kann auch mittels der Masse des Körpers und der auf ihn einwirkenden translatorischen Kräfte ermittelt werden, indem man $\vec{F} = ma$ nach der Beschleunigung umstellt. Die Einheit der Beschleunigung ist $\frac{m}{s^2}$.

3.4 Drehdynamik

Die Drehdynamik (eigentlich schlicht nur Dynamik) ist wie die Kinematik ein Teilgebiet der klassichen Mechanik und befasst sich mit der Wirkung von Kräften

3.4.1 Orientierung

Um innerhalb der Simulation die Orientierung des Hubschraubers beschreiben zu können werden Quaternionen eingesetzt. Diese bieten eine kompakte Darstellung und durch ihre Algebra sind sie prädestiniert für diesen Zweck. Zudem sind sie im Vergleich zu Rotationsmatrizen numerisch stabiler und müssen seltener orthogonalisiert werden. Eine genaue Definition und Grundlagen im Umgang mit Quaternionen können [Eberly, 2015] entnommen werden. Es gibt eine ganze Reihe von Vorteilen gegenüber den

anderen Möglichkeiten eine Rotation zu repräsentieren. Die Darstellung mittels einer Rotationsmatrix M ist zwar mathematisch nicht sonderlich problematisch, allerdings hat man für eine einzige Rotation mit drei Freiheitsgraden neun Matrixeinträge. Zudem muss die Rotationsmatrix nach [Eberly, 2015] immer mal wieder orthongonalisiert werden. Dies kann mit dem Gram-Schmidt-Verfahren durchgeführt werden, ist allerdings bei neun Werten in der Rotationsmatrix aufwändiger als die Normalisierung eines Quaternions. Des Weiteren ist die Darstellung mittels der Matrix nicht intuitiv. Euler- beziehungsweise Kardanwinkel haben den großen Nachteil, dass bei bestimmten Akkumulationen von Rotationen der so genannte Gimbal-Lock auftritt. Dadurch verliert die Representation einen Freiheitsgrad und alle folgenden Rotationen sind im Anschluss möglicherweise nicht mehr berechenbar. Diese Nachteile treten bei der Darstellung von Rotationen durch Quaternionen nicht auf. Zwar muss auch ein Quaternion hin und wieder Orthognonalisiert werden, aber weitaus seltener als eine Rotationsmatrix. Allgemein wird ein Rotationsquaterion wie im einleitenden Abschnitt durch

$$q = w + xi + yj + zk \quad mit \quad x, y, z, w \in \mathbb{R} \tag{3.4}$$

dargestellt. i, j, k sind imaginäre Einheiten. Ein Quaternion mit dem Betrag Eins wird als Einheitsquaternion bezeichnet. Damit ist die Orientierung des Hubschraubers zu jedem Zeitpunkt eindeutig festgelegt.

3.4.2 Winkelgeschwindigkeit

Die Winkelgeschwindigkeit ist als die zeitliche Änderung der Orientierung definiert und wir durch

$$\omega = \frac{\Delta\varphi}{\Delta t} \tag{3.5}$$

angegeben. Da der Drehwinkel φ in Radiant angegeben wird ist die Einheit der Winkelgeschwindigkeit $\frac{rad}{s}$ was bei der Implementation beachtet werden muss. ω zeigt dabei in die Richtung der Rotationsachse und der Betrag gibt die Geschwindigkeit an. Ziel soll es später sein, auf der Grundlage der ermittelten Drehkräfte, die Winkelgeschwindigkeit zu bestimmen und daraus die neue Orientierung zu extrapolieren.

3.4.3 Drehmomente

Für die Hubschraubersteuerung sind im Groben drei sogenannte Hauptdrehmomente von Bedeutung. Diese werden mit M_x, M_y und M_z bezeichnet und beziehen sich entsprechend auf die Roll-, Nick-, oder Gierachse des Hubschraubers. Das Drehmoment spielt bei der Rotation eine ähnliche Rolle wie bei der Translation die Kraft. Für einen gegebenen Körper existiert

ein Ortsvektor $\vec{r}$, der den Angriffpunkt der Drehkraft angibt. Im Folgenden mit $\vec{M}$ bezeichnet kann das Drehmoment mittels

$$\vec{M} = \vec{r} \times \vec{F} \tag{3.6}$$

über das Kreuzprodukt berechnet werden. Wirkt die Kraft $\vec{F}$ im rechten Winkel zum Hebelarm l, so kann die Gleichung aus 3.6 umgeschrieben werden, sodass

$$\vec{M} = l \cdot \vec{F} \tag{3.7}$$

ist. l gibt dabei den Abstand vom Masseschwerpunkt zur wirkenden Kraft durch $l = \mid \vec{r} \mid$ an [Tipler et al., 2015]. Das Drehmoment wird dabei häufig in der Einheit Nm angegeben. Es kommt aber auch vor, dass anstelle von Meter ein Bruchteil dessen als Basis gewählt wird, wenn es sich um kleinere Maßstäbe handelt.

3.4.4 Trägheitsmomente

Je nach Gestalt des Starrkörpers hat dieser einen gewissen Widerstand hinsichtlich der Änderung seiner Bewegung. Bei der Translation wurde in Abschnitt 3.2.2 erwähnt, dass die Masse m eines Körpers der Widerstand zu translatorischen Bewegungsänderungen ist. Ein Pendant hinsichtlich der Rotation ist das Trägheitsmoment

$$I = \rho \int_V r_\perp^2 dV \tag{3.8}$$

mit ρ für die Dichte des Körpers und r^2 als die Strecke senkrecht zur Rotationsachse. In Gleichung 3.8 geht man von einer homogenen Massenverteilung aus, was es ermöglicht, die Dichte vor das Integral zu schreiben. Je nach Rotationsachse sind die Trägheismomente verschieden. Für einen Körper lassen diese sich in einem Trägheitstensor I zusammenfassen. Da wir eine Bounding-Box als Körper für die Simulation annehmen, werden auf Basis dieser die Trägheitsmomente zusammengefasst:

$$I = \begin{pmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{pmatrix} \quad I_{Box} = \begin{pmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{pmatrix} \tag{3.9}$$

Aufgrund von Symmetrieeigenschaften fallen bei I_{Box} alle Volumenintegrale mit verschiedenen Achsen weg, sodass nur die Werte auf der Hauptdiagonalen stehen bleiben. Für die übrigen Einträge folgt

$$I_{xx} = \int_R y^2 + z^2 dm \quad I_{yy} = \int_R x^2 + z^2 dm \quad I_{zz} = \int_R x^2 + y^2 dm \tag{3.10}$$

mit x, y, z als Kantenlänge der Bounding-Box. Als Ergebnis folgen die unten aufgeführten Hauptträgheitsmomente und werden so dann auch numerisch in der Simulation implementiert:

$$I_{xx} = \frac{M}{12}(b^2 + c^2) \quad I_{yy} = \frac{M}{12}(a^2 + c^2) \quad I_{zz} = \frac{M}{12}(a^2 + b^2) \tag{3.11}$$

3.4.5 Drehimpuls

Der Drehimpuls ist eine Analogie zum Impuls der Translation. Um einen Körper in Rotation zu versetzen ist ein Drehmoment erforderlich, um dem Körper einen Drehimpuls beizubringen. Ein veralteter Begriff für Drehimpuls ist Drall. In [Tipler et al., 2015] wird der Drehimpuls wie folgt definiert:

$$L = r \times p \tag{3.12}$$

Hierbei bezeichnet L den Drehimpuls, r ist der Abstand des betrachteten Masseteilchens vom Schwerpunkt und p dessen linearer Impuls mit $p = mv$. Wichtig ist an dieser Stelle der Zusammenhang zwischen Drehmoment und Drehimpuls. Durch

$$M_{ext} = \frac{dL}{dt} \tag{3.13}$$

ist die Verbindung zwischen einem externen Drehmoment und Drehimpuls als zeitliche Änderung des Drehimpulses gegeben. Tipler et al. beschreibt die Rotation eines symmetrischen starren Körpers um dessen Z-Achse mit $L_z = I_z\omega$. Hier ist I_z das Trägheitsmoment bezüglich der Z-Achse des Starkörpers. Daraus folgert er mit

$$M_{ext} = \frac{dL_z}{dt} = \frac{d}{dt}(I_z\omega) = I_z\alpha \tag{3.14}$$

das Produkt aus Winkelbeschleunigung α und dem entsprechenden Trägheitsmoment als Entsprechung für das externe Drehmoment. Dieser Zusammenhang wird später in der Simulation dazu ausgenutzt, um aus den Drehmomenten die Winkelbeschleunigungen und damit einhergehend die Winkelgeschwindigkeiten in jedem Schritt zu bestimmen.

4 Simulationsmodell

4.1 Randbedingungen

An dieser Stelle seien eine Reihe von Idealisierungen aufgeführt, die im Rahmen dieser Arbeit gemacht wurden. Die Simulationsdomäne befindet sich in einem windstillen Feld, sodass lediglich der Hubschrauber selbst mit seiner Bewegung und den Rotoren entsprechende Blattanströmungen erzeugt. Eine Simulation beziehungsweise Wechselwirkung von Luft ist denkbar, erhöht aber aufgrund der Kompressibilität die Komplexität der Berechnungen. Eine weitere Idealisierung ist die Berechnung des Trägheitstensors. An dieser Stelle wird der Tensor auf Grundlage der Bounding Box um die komplette Hubschraubegeometrie ermittelt. Es gibt grobe Schätzwerte für die Widerstandsverluste des Rumpfes, da diese in der Regel aufwändig mittels CFD-Simulation (Computational Fluid Dynamics) berechnet werden oder experimentell ermittelt müssen und daher in der Regel nicht echtzeitfähig sind. Desweiteren sind Folgende aerodynamische Effekte ausser Acht gelassen worden:

1. Schräganströmung
2. Rückanströmung
3. Machzahleffekte
4. Blatt-Wirbel-Interaktion
5. Rotor-Rumpf-Interaktion
6. Strömungsablösung, sofern nicht in den Daten der Polardiagramme berücksichtigt
7. Dynamischer Strömungsabriss

Weiter wird auf eine selbstentwickelte Kollisionserkennung verzichtet. An dieser Stelle kommt die integrierte Erkennung der 3D-Engine zum Einsatz, welche zu großen Teilen auf Nvidia PhysX beruht. In Abschnitt 5.3.2 wird kurz erläutert, wie eine Kollisionserkennung erfolgt.

4.2 Numerische Lösungsverfahren

Bei der Lösung der Bewegungsgleichungen aus den Abschnitten 3.3 und 3.4 handelt es sich üblicherweise um Probleme bei der Lösung von gewöhnlichen Differentialgleichungen (ODEs). Verkürzt ausgedrückt handelt es sich bei ODEs um eine Gleichung mit einer Variablen, die eine gesuchte Funktion formuliert. Diese Funktion gilt es zu ermitteln, was meist analytisch nicht möglich ist und daher auf numerische Lösungsansätze zurück gegriffen werden muss. Typische ODEs sind der Gestalt, dass mit

$y\prime = g(x) \cdot h(y)$ auch Ableitungen der Funktion selbst auftreten. Es soll ein Überblick über ein einfaches numerisches Verfahren gegeben werden, sowie dessen Verbesserung, welches auch im Rahmen dieser Arbeit zum Einsatz kommt. Da eine Simulation zu unterschiedlichen Zeitschritten stattfindet und man mit einem Computer nicht beliebig genau werden kann, muss dieser Sachverhalt bei der Wahl der Algorithmen mit in Betracht gezogen werden.

4.2.1 Euler-Verfahren

Das Euler-Verfahren (oder auch explizites Euler-Verfahren) ist ein verhältnismäßig einfaches Verfahren zur Lösung von ODEs. Die Rechenvorschrift für das Euler-Verfahren sieht auf dem ersten Blick trivial aus (ist es zugegebener Maßen auch), hat allerdings ein paar Konsequenzen für die Stabilität der Simulation. Pro Simulationsschritt ist man daran interessiert aus einem gegebenen Objektzustand zum Zeitpunkt t den Zustand für $t + \Delta t$ zu berechnen. Die Gleichungen für das Eulerverfahren sind durch

$$v(t + \Delta t) = v(t) + (\Delta t)v\prime(t) \tag{4.1}$$

$$x(t) = x(t) + \Delta t v(t) \tag{4.2}$$

gegeben. Es wird also versucht, die Geschwindigkeit v für den nächsten Zeitschritt dadurch zu berechnen, dass man die Tangentensteigung zum Zeitpunkt t wählt und dann linear Interpoliert um den Wert für die Geschwindigkeit für $t + \Delta t$ zu erhalten. Wie aus Abbildung 12 ersichtlich ist, handelt man sich ganz besonders bei beschleunigten Bewegungen numerische Fehler ein. Diese Fehler sind nicht zu unterschätzen und gefährden die Stabilität der Simulation zum Teil enorm. Daher wird das Euler-Verfahren im folgenden Abschnitt leicht modifiziert und bekommt so stabilere Eigenschaften [Dahmen and Reusken, 2008].

4.2.2 Semi-implizites Euler-Verfahren

Um eine bessere numerische Stabilität zu erhalten und den Fehler zwischen dem berechneten $t+\Delta t$-Wert im Vergleich zum tatsächlichen Funktionsverlauf zu minimieren wurde das Semi-implizite Euler-Verfahren entwickelt. Wie bereits angesprochen handelt es sich dabei um eine Modifikation des in Abschnitt 4.2.1 gezeigten Verfahrens. Erreicht wird dies dadurch, dass zuerst die neue Geschwindigkeit für den kommenden Zeitschritt berechnet wird und auf Grundlage dessen eine Aktualisierung der Position und Orientierung erfolgt. Der Vorteil dieses Ansatzes liegt auf der Hand. Wurde vorher die Interpolation mittels eines, umgangssprachlich ausgedrückt, veralteten Wert berechnet, so ist der Fehler bei der Aktualisierung dessen vor der Interpolation kleiner. Dennoch gilt Vorsicht, denn auch das Semi-implizite Euler-Verfahren ist nur eine einfache Approximation. Treten sehr

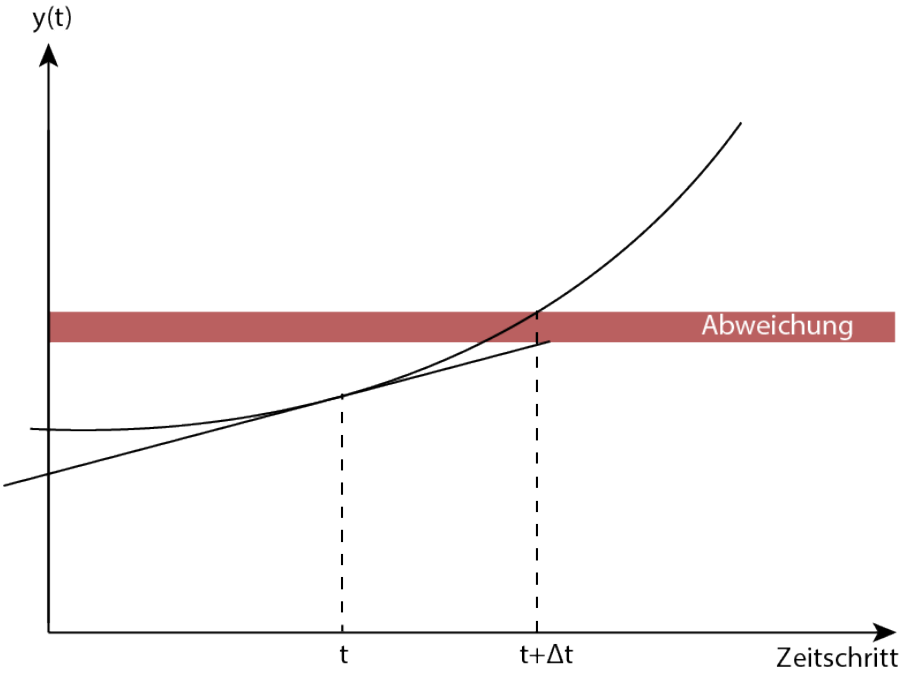

Abbildung 12: Integrationsschritt mit dem expliziten Euler-Verfahren. Gut zu erkennen ist die Fehlerentwicklung nach einem Rechenschritt. Dies tritt besonders stark bei Funktionen mit hohen Differenzen in der Tangentensteigung auf.

große Differenzen in den Zuständen zwischen zwei Zeitschritten auf, ist auch der Fehler entsprechend groß. Daher ist man bemüht den Zeitschritt und damit das Δt möglichst klein zu halten. Zum Teil werden in Simulationen, die hohe Präzision erfordern, daher mehrere hundert mal pro Sekunde die Physik-Simulation gerechnet und danach ein neuer Render-Frame generiert. Vorweg gegriffen sind mit der Unreal Engine bei dem einfachen grafischen Szenario Zeitinkremente im mittleren Bereich von Mikrosekunden aufgetreten. Benötigt man zusätzliche Stabilität ohne hochpräzise Ergebnisse zu fordern, kann man auch Algorithmen wählen, die auf Leapfrog, Verlet-Velocity oder dem Verfahren von Runge und Kutta basieren.

4.3 Simulationsablauf

In Abbildung 17 ist ein Durchlauf dargestellt. Zu Beginn erfolgt die Initialisierung der Simulationsparameter. Die Initialisierung der Engine selbst und daran angeschlossenen Systeme zum Rendering sind an dieser Stelle vernachlässigt. Danach werden von der Engine die Steuereingaben entgegen genommen und entsprechen umgerechnet und interpoliert. Darauf fol-

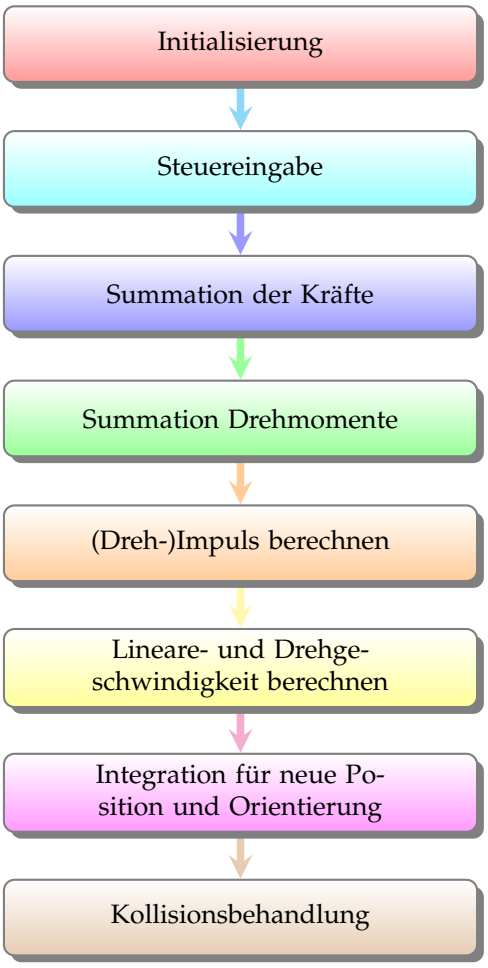

Abbildung 13: Simulationsschritte im Durchlauf

gend werden alle Kräfte berechnet und aufsummiert. Handelt es sich um Drehkräfte, so werden diese gesondert aufsummiert. Mittels numerischer Integration folgen aus den vorher summierten Kräften die Impulse p und L. p wird durch die Masse dividiert und ergibt die neue translatorische Geschwindigkeit. Aus L folgt durch die Multiplikation mit dem inversen Trägheitstensor die neue Winkelgeschwindigkeit ω. Eine erneute Integration von v und ω ergibt die neue Position beziehungsweise die neue Orientierung. Abschließend findet eine einfache Kollisionsbehandlung statt, die sicherstellt, dass der Hubschrauber nicht durch das Terrain-Mesh fällt und die Geschwindigkeit entsprechend auf Null gesetzt wird.

5 Implementation

5.1 Engine

Als Basis dient dieser Simulation die Unreal Engine (kurz: Unreal) von Epic. Verschiedenste Konzepte sprachen für die Wahl dieser Engine, unter anderem C++ als verwendete Programmiersprache. Durch die Benutzung der Unreal Engine werden visuelle Effekte, die das Rendering betreffen, erleichtert und der Fokus bei der Entwicklung auf der physikalischen Simulation beibehalten. In den kommenden Abschnitten wird die Implementation in der Unreal Engine erklärt und Klassen sowie Algorithmen am Detail erläutert. Ein wichtiges Detail sei an dieser Stelle erwähnt. Wie in den vorherigen Gleichungen und physikalischen Gesetzmäßigkeiten wird konsequent in SI-Einheiten gerechnet. Intern rechnet die Unreal Engine mit eigenen Units, welche sich auf 10cm beziehen, sprich eine Engine Unit entspricht 10cm. Daher werden alle physikalischen Berechnungen in SI-Einheiten durchgeführt und zum Ende hin findet eine Umrechnung in die genannten Engine Units statt. Das ist die einzige Besonderheit. Winkel sind in der Regel nicht Engine-spezifisch und daher nicht allgemein gültig.

5.2 Pawn Klasse

Das Grundgerüst des Hubschraubers ist mittels der von Unreal zur Verfügung gestellten `Pawn`-Klasse realisiert. Diese Klasse repräsentiert ein vom Spieler steuerbares Objekt in der Spielewelt. Darauf aufbauend, können verschiedene Szenenobjekte wie Meshes, Kameras oder Kollisionsprimitive in eine Art Szenengraphen an dieses Objekt angefügt werden. In Abbildung 14 erkennt man die einzelnen Komponenten für das Renderning, sowie die `MovementComponent`-Klasse.

5.2.1 Static Mesh Komponente

Die 3D-Meshes des Rumpfes und der beiden Rotoren wurde als `StaticMesh`-Komponente an das `Pawn`-Objekt angefügt. Dazu wurden sie mit den erforderlichen lokalen Transformationen an die `RootComponent` verankert, sodass diese relativ dazu im Szenegraphen hängen.

5.2.2 Controller-Steuerung

Die Controller-Eingaben werden von der Engine als so genannte Action- und Axismappings zur Verfügung gestellt. Actionmappings beziehen sich dabei auf Tastendrücke und Axismappings auf stetige Eingaben wie etwa die X-Achsenposition des Joysticks. Diese Axismappings geben einen Float-Wert im Bereich von 0 bis 1 an die im Setup-Code angegebene Callback-Methode weiter. Zu Beginn eines Renderdurchlaufs, fragt die Engine alle

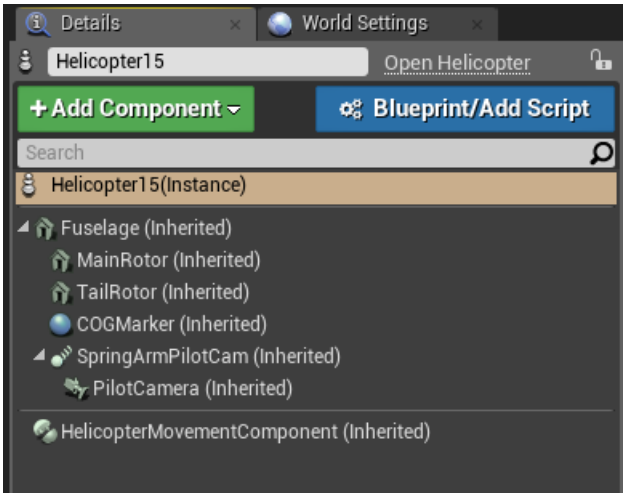

Abbildung 14: Editorsicht der Hubschrauber-Komponenten

Controller oder Tastatureingaben, die über entsprechende Mappings verfügen, ab und gibt diese dann an die entsprechenden Methoden weiter. Für die X-Achse, was der Nick-Achse entspricht, wird dabei dann die Funktion `CyclicElevation(float val)` aufgerufen.

```
void AHelicopter::CyclicElevationInput(float Value)
{
  m_cyclicElevation = Value;
  if (HelicopterMovementComponent && (HelicopterMovementComponent
    ->UpdatedComponent == RootComponent))
  {
    HelicopterMovementComponent->AddInputVector(
    GetActorForwardVector() * Value);
  }
}
```

Listing 1: Callback und Weitergabe der Steuereingabe

Die Methode `AddInputVector` reicht die Eingabe von der `Helicopter`-Klasse an die `MovementComponent` weiter. In dieser Klasse wird die Steuereingabe entgegengenommen und entsprechend der Hauptrotorsteuerwinkeln interpoliert, wie in Listing 2 zu sehen.

```
float UHelicopterMovementComponent::CyclicElevationInput(FVector
    input)
{
  const float cyclicElevationValue = input.X;
  const auto val = (1 - cyclicElevationValue) * MainRotor->
    minCyclicElevation + cyclicElevationValue * MainRotor->
    maxCyclicElevation;
  return val;
}
```

Listing 2: Interpolation des Nick-Steuerwinkels

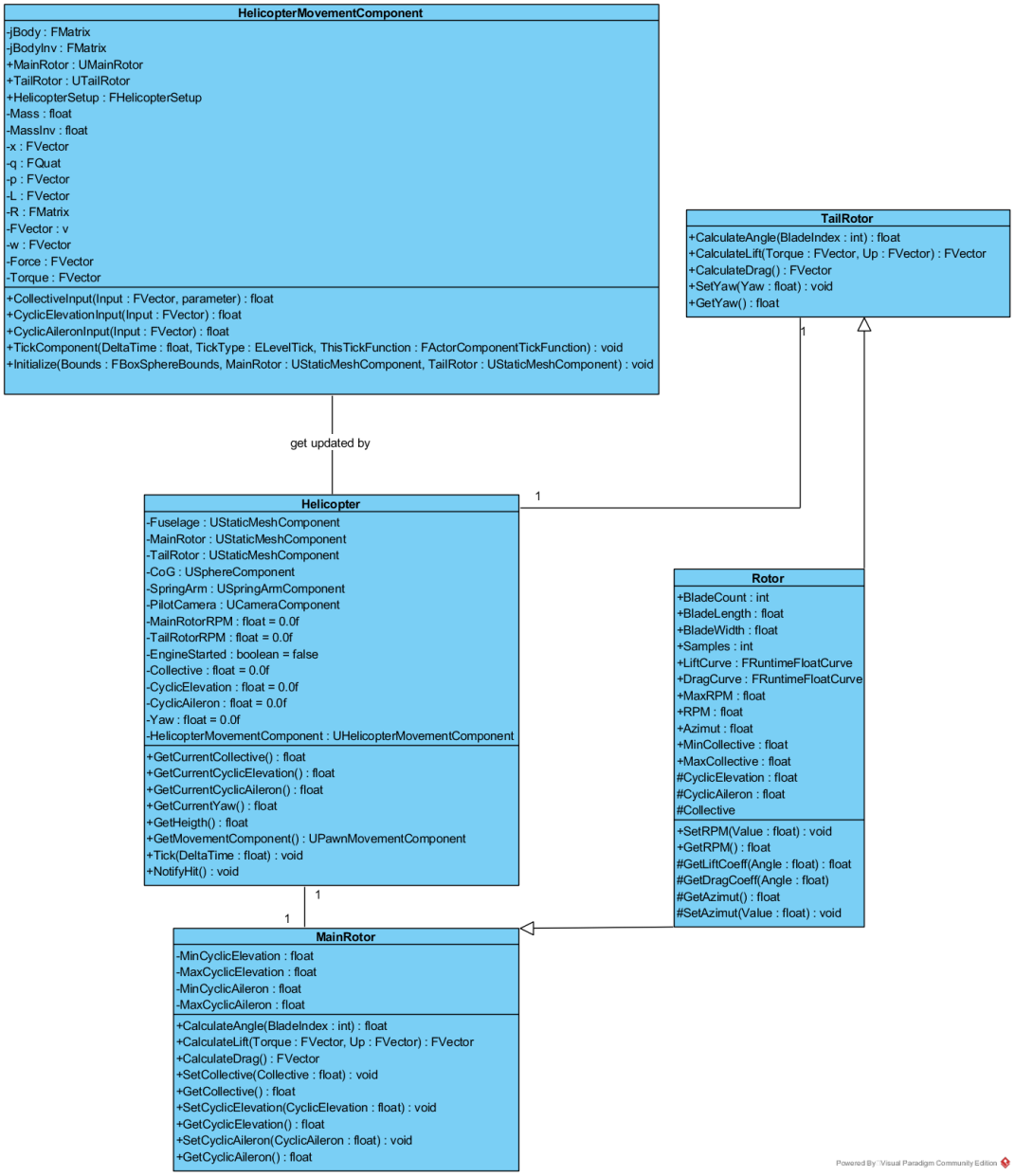

Abbildung 15: UML-Klassendiagramm der Simulationsumgebung

5.2.3 Camera Komponente

Direkt unterhalb der `Pawn`-Komponente ist eine Kamera über eine `SpringArm`-Komponente am 3D-Mesh verankert und wird dem entsprechend bei der Bewegung nachgeführt.

5.3 Movement-Component Klasse

Innerhalb der `Pawn`-Klasse befindet sich die von `MovementComponent` abgeleitete Klasse `HelicopterMovementComponent`. Folgende Aufgaben übernimmt diese Klasse:

1. Entgegennahme der Steuereingaben von Tastastur oder Joystick.
2. Berechnung der Steuerwinkel θ der jeweiligen Rotorblätter.
3. Berechnung des Rotorauftriebs durch die `MainRotor`-Klasse.
4. Berechnung des Drehmomentes durch die `TailRotor`-Klasse.
5. Summation der translatorischen Kräfte und Integration.
6. Summation der rotatorischen Kräfte und Integration.
7. Update des Ortsvektors $\vec{x}$ und der Orientierung q.

Vom Pawn wird die Methode `TickComponent` jedes mal aufgerufen, wenn ein Update ansteht. Innerhalb dieser Methode finden dann alle Berechnungen statt und der Pawn wird mit seiner Position und Orientierung über den Methodenaufruf aktualisiert.

5.3.1 Initialisierung

Ganz zu Beginn der Simulation muss der Anfangszustand des Hubschraubers initialisiert werden. Dazu zählen offensichtlich Position $\vec{x}$ und Orientierung q, aber auch dessen Geschwindigkeit v, die Rotationsgeschwindigkeiten ω. Darüber hinaus werden alle anderen Parameter wie Rotordrehzahlen und dergleichen auf Null gesetzt. Die Startposition leitet sich direkt aus der Position der RootComponent des Pawn ab. Den ersten Teil der Initialisierung ist in Listing 3 zu sehen.

```
1 UHelicopterMovementComponent::UHelicopterMovementComponent()
2 {
3   MainRotor = CreateDefaultSubobject<UMainRotor>(TEXT("
      MainRotorInstance"));
4   TailRotor = CreateDefaultSubobject<UTailRotor>(TEXT("
      TailRotorInstance"));
5   EngineStarted = false;
6   HelicopterSetup.Mass = 1455.0f;
7   mass = HelicopterSetup.Mass;
```

```
  massInv = 1.f/mass;
  OffsetTailRotor = 0.f;

  jBody = FMatrix::Identity;
  jBodyInv = FMatrix::Identity;

  x = FVector(0.f);
  q = FQuat(); //init below
  p = FVector(0.f);
  L = FVector(0.f);
}
```

Listing 3: Erster Teil der Initialisierung

Diese Position kann interaktiv im Editor festegelegt werden. Auf gleiche Weise wird die Rotation festgelegt und initialisiert. Dabei wird auch schon einmal die Rotationsmatrix R zusätzlich zur Quaternionen-Repräsentation berechnet, um später dann darauf basierend den inversen Trägheitstensor und den Drehimpuls zu ermitteln. Des Weiteren wird auch die Masse festgelegt. Überlichweise würde dies mit einer Schleife über alle Massenelemente ablaufen und die Gesamtmasse dementsprechend akkumulieren. In diesem Fall aber lässt sich die Masse des Hubschraubers aber einfach über den Editor manuell festlegen. Ebenso wird der Massenschwerpunkt (center of gravity - CoG) manuell im Editor angepasst. Dies ist über eine `CollisionSphere` realisiert, die lediglich einer Debug-Visualisierung dient und dem Benutzer einen 3D-Gimbal zum Verschieben anzeigt. Danach wird der Trägheitstensor I bestimmt, da dieser sich im lokalen Koordinatensystem während der Simulation nicht ändert und dementsprechend problemlos vorberechnet werden kann. Dies geschieht wie in Listing 4 über die Abmessung der Bounding-Box, die über die Engine-eigene Struktur `FBoxSphereBounds` geliefert wird.

```
void UHelicopterMovementComponent::InitializeTensor(
    FBoxSphereBounds Bounds)
{
  FVector Min = Bounds.GetBox().Min;
  FVector Max = Bounds.GetBox().Max;

  float mass = HelicopterSetup.Mass;

  float Length = Max.X - Min.X;
  float Height = Max.Z - Min.Z;
  float Width = Max.Y - Min.Y;

  //convert unreal units to meters
  Length /= 100.f;
  Height /= 100.f;
  Width /= 100.f;

  float xx = mass * (((Height * Height) + (Length * Length)) /
    12.0f);
```

```
18    float yy = mass * (((Width * Width) + (Length * Length)) / 12.f)
        ;
19    float zz = mass * (((Width * Width) + (Height * Height)) / 12.f)
        ;
20
21    jBody.SetIdentity();
22    jBody.M[0][0] = xx;
23    jBody.M[1][1] = yy;
24    jBody.M[2][2] = zz;
25    jBodyInv= jBody.Inverse();
26  }
```

Listing 4: Initialisierung Trägheitsmomente

Im Anschluss lässt sich dann der inverse Trägheitstensor bestimmen mit $I^{-1} = RI_{Body}R^T$. Ebenfalls kann dann der Drehimpuls mit $L = I\omega$ initial festgelegt werden. Da ω am Anfang der Simulation auf Null gesetzt wurde ist auch der Drehimpuls für diesen initialen Zeitschritt gleich Null. Dies geschieht wie in 5 dargestellt. Zur Sicherheit wird nochmal die Membervariable `UpdateComponent` auf Null-Pointer überprüft, um Laufzeitfehler auszuschließen.

```
1  void UHelicopterMovementComponent::Initialize(const
       FBoxSphereBounds Bounds, const UStaticMeshComponent *
       MainRotor, const UStaticMeshComponent * TailRotor)
2  {
3    InitializeTensor(Bounds);
4    InitializeRotors(MainRotor, TailRotor);
5
6    if(UpdatedComponent)
7    {
8      q = UpdatedComponent->RelativeRotation.Quaternion();
9      R = FQuatRotationMatrix::Make(q);
10     jInv = R * jBody * R.GetTransposed();
11     v = massInv * p;
12     w = jInv.TransformVector(L);
13   }
14
15   ...
16 }
```

Listing 5: Initialisierung Trägheitsmomente

5.3.2 Kollisionserkennung

Bei der Kollisionserkennung wird auf die von in Unreal Engine enthaltenen Funktionen zurückgegriffen. Von der Engine wird eine `HitResult` Struktur bereitgestellt, die alle nötigen Informationen wie Kontaktpunkt, Kontaktnormale und Zeitpunkt angibt. Zurückgeliefert wird das Ergebnis von `SafeMoveUpdatedComponent`, welche versucht, die neue Translation und Rotation auf das von der MovementComponent gesteuerten Objekt anzuwenden. Exemplarisch ist dies in Listing 6 dargestellt.

Abbildung 16: Fuselage-Mesh mit Bounding-Box (grün)

```
// If we bumped into something, try to slide along it
if (Hit.IsValidBlockingHit())
{
  SlideAlongSurface(Velocity, 1.f - Hit.Time, Hit.Normal, Hit);

  if(Hit.ImpactNormal == FVector(0.f, 0.f, 1.f) && Velocity.Z < 0.
    f)
  {
    // Do not fall through plane mesh
    Velocity.Z = 0.f;
  }
}
```

Listing 6: Kollisionsbehandlung

Kommt es dabei zu einer Kollision, wird intern `ResolvePenetration` aufgerufen und das Objekt aus der Kollision raus bewegt. Hauptsächlich finden Kollisionen mit dem Terrain-Mesh statt, welches in dieser Simulation als einfache Ebene angelegt ist. Die Pawn-Meshes, beschrieben in Abschnitt 5.3.2, haben Collisions-Shapes, wie sie in Abbildung 16 zu erkennen sind. Diese sind im Mesh-Editor von Unreal erzeugt. Dabei gibt es die Wahl zwischen simplen Boundingboxen oder komplexen und hochdefinierten k-Polytopen. Zur Berechnung der Trägheitsmomente wird allerdings eine einfache Bounding-Box angenommen, die ebenfalls von der Engine berechnet wird.

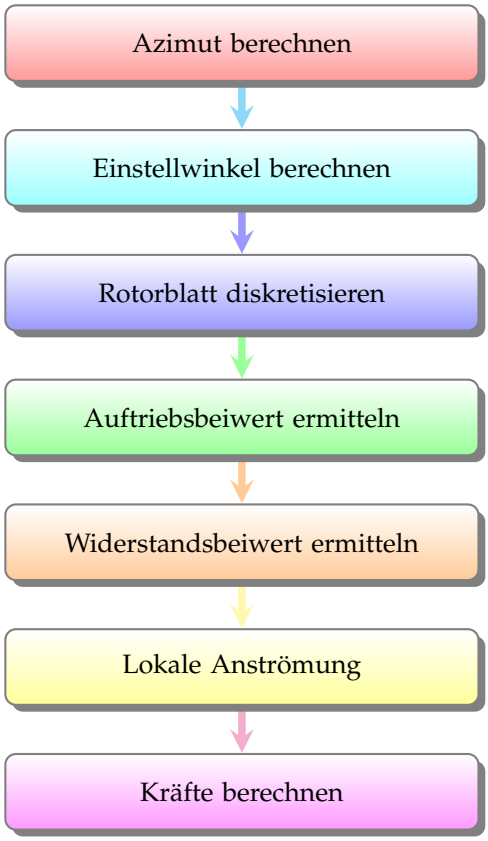

Abbildung 17: Ablauf zur Berechnung der Auftriebskraft und der Widerstandskraft

5.3.3 Main Rotor

Wie aus dem UML-Diagramm in Abbildung 15 zu erkennen ist, beinhalten die Rotorklassen überwiegend ähnliche Funktionalität. Dennoch wurden zwei separate Klassen entworfen, welche ein gemeinsames Interface implementieren. Triviale Parameter des Hauptrotors sind dessen Blattanzahl N_b, Blattlänge y und die Umdrehungsgeschwindigkeit Ω. Zusätzlich dazu kommen aber noch Einstellwerte, die wichtig in Hinblick auf die Steuerung sind. Die in Abschnitt 5.2.2 beschriebenen Eingabewerte für die Axismappings werden an den Hauptrotor für die Steuerung weitergegeben. Innerhalb der Klasse erfolgt dann die lineare Interpolation mit den jeweiligen Min-/Max-Werten für die kollektiven und zyklischen Blattanstellwinkel. Um später die Kräfte am Hauptrotor korrekt ermitteln zu können ist die Berechnung der Steuereingaben pro Rotorblatt notwendig.

Listing 7 zeigt dabei, wie der Ablauf ist. Es wird über `BladeIndex` iteriert und so für jedes Rotorblatt der individuelle Einstellwinkel berechnet.

Dabei wird der Rotorkreis in 90 Grad aufgeteilt und entsprechend Gleichung 2.6 der dazugehörige Winkel zurückgeliefert.

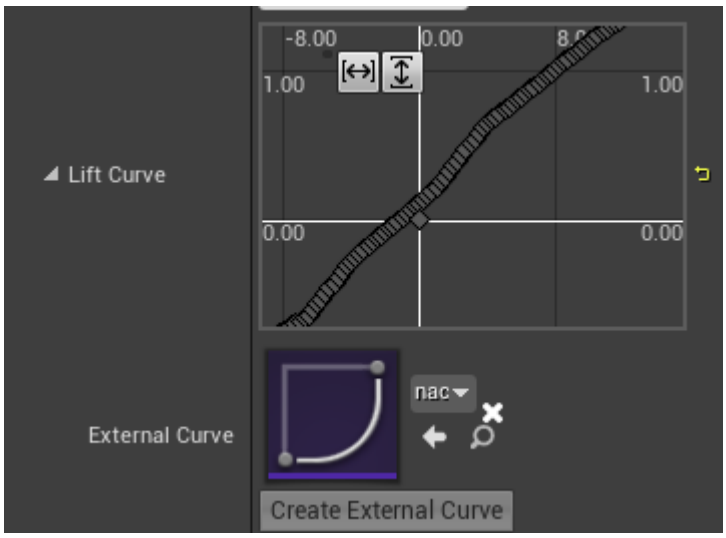

Abbildung 18: Auftriebsbeiwerte C_l im Editor für den Hauptrotor

```
float UMainRotor::CalculateAngle(const int BladeIndex)
{
  const float bladeAzimut = fmod(Azimut + BladeIndex * 90.f, 360);
  const auto AzimutRadians = FMath::DegreesToRadians(bladeAzimut);
  float bladeAngle = collective + FMath::Cos(AzimutRadians) *
    cyclicAileron + FMath::Sin(AzimutRadians) * cyclicElevation;
  return bladeAngle;
}
```

Listing 7: Berechnung der Gesamtkontrolleingabe

Anschließend findet, wie in Code-Listing 8 dargestellt, die Berechnung der Auftriebskraft am Hauptrotor dar. Für jeden Simulationsschritt wird `CalculateLift` aufgerufen und liefert die gesamte Auftriebskraft aller Rotorblätter des Hauptrotors in Newton zurück. Die äußere Schleife läuft dabei über alle Rotorblätter und berechnet den jeweiligen Azimutwinkel, wie in Listing 7 erläutert wurde. Darauf folgt die Abfrage des Auftriebsbeiwertes. Dieser wird von der `FRuntimeFloatCurve` aus Abbildung 18 zurückgeliefert und interpoliert. Damit steht der Auftriebsbeiwert für das Rotorblatt und den Simulationsschritt fest und ändert sich nicht mehr. Auf Grundlage der Blattelementtheorie aus Abschnitt 2.5 müssen erst die lokalen Anströmungsgeschwindigkeiten berechnet werden. Dafür existiert eine globale Variable `Samples`, die nach [Wall, 2015] zwischen 10 und 50 gewählt werden kann, um die in Abschnitt 2.5.1 aufgestellten Gleichungen näherungsweise zu lösen.

```cpp
float UMainRotor::CalculateLift()
{
  float lift = 0.f;

  for(int i = 0; i < BladeCount; i++)
  {
    const float angle = CalculateAngle(i);
    const float liftCoeff = GetLiftCoeff(angle);
    const float segmentWidth = BladeLength / Samples; // Each segment for calculation is 10% of total blade length

    for(int j = 0; j < Samples; j++)
    {
      const float segmentVelocity = 2.f * PI * (RPM / 60.f) * segmentWidth * j + (segmentWidth/2.f) ;
      lift += 0.5f * liftCoeff * 1.2041f * BladeWidth * segmentWidth * FMath::Pow(segmentVelocity, 2);
    }
  }
  return lift;
}
```

Listing 8: Berechnung der Auftriebskraft

In Abhängikeit von der Anzahl der Samples wird das Rotorblatt in gleich große Abschnitte unterteilt. In der Mitte eines jeden Abschnittes liegt die Segment-Geschwindigkeit, gegeben durch $\omega = 2 \cdot \pi f$ und $v = \omega \cdot r$, an. Mit dieser lokalen Geschwindigkeit kann schlussendlich die Auftriebskraft berechnet werden kann. Dieser Vorgang läuft über alle Segmente und alle Rotorblätter. Zusätzlich wird gespeichert, bei welchem Azimutwinkel die entsprechenden Auftriebskräfte auftreten. Mit dieser Information können die Drehmomente in X-Richtung und Y-Richtung errechnet werden, was später wichtig wird um die Steuerbarkeit zu den Seiten zu gewährleisten. Die Idee dahinter ist ähnlich dem Mischen der Steuereingaben aus Abschnitt 2.3.4. Der Rotorkreis wird dabei gedanklich in viertel aufgeteilt, sodass für einen gegebenen Azimutwinkel die Funktionen für Sinus und Cosinus dort ihr Maximum, respektive Minimum haben. Die Berechnung soll einmal für die Pitch-Bewegung, also das Nicken mit der Nase nach oben und unten verdeutlicht werden.

Intuitiv könnte man davon ausgehen, dass die Rotorblätter vorne und hinten (gesehen in X-Richtung) den größten Beitrag zur Nick-Bewegung eines Hubschraubers beitragen. Nämlich genau dann, wenn asymmetrischer Auftrieb erzeugt wird, sprich das vordere Rotorblatt erzeugt mehr Auftrieb als das hintere und umgekehrt. Folglich hebt sich die Hubschraubernase. Dabei ist allerdings die Kreiselpräzession ausser acht gelassen worden [Tipler et al., 2015][Johnson, 1994]. Dieser Sachverhalt bewirkt, dass die Kraftwirkungen für den Auftrieb am Hauptrotor um 90 Grad verschoben sind. Konkret bedeutet dies, dass ein Rotorblatt mit einem Azimutwinkel von 0

Grad die Kraftwirkung bei Azimut mit 90 Grad zeigt. Dies lässt sich über den gesamten Rotorkreis fortführen. Demnach sind also die Rotorblätter links und rechts, bei den Azimutwinkeln 90 und 270 Grad, dafür verantwortlich eine Nickbewegung auszuführen. Um jetzt die Beiträge der Rotorblätter zum jeweiligen Nick- oder Rollmoment zu berechnen, werden die in Algorithmus 1 aufgeführten Schritte ausgeführt. Zu Beginn initiali-

Algorithm 1 Nick-Steuerung

```
1:  procedure CalcNickContribution(Blades)
2:      leftBlade ← blade for left conribution
3:      rightBlade ← blade for right conribution
4:      for each Rotorblade in Blades do
5:          Azimut ← Azimut angle of Rotorblade
6:          contrib ← sin(Azimut)
7:          if contrib < leftBlade then
8:              leftBlade ← Rotorblade
9:          else if contrib > rightBlade then
10:             rightBlade ← Rotorblade
        return leftBlade, rightBlade
```

siert man die beiden Variablen *leftBlade* und *rightBlade* mit Standardwerten. Danach folgt eine Schleife über alle Rotorblätter, die zuerst den aktuellen Azimutwinkel des entsprechenden Rotorblattes abfragt und dessen Sinus berechnet. Sinus deshalb, weil dieser bei den Winkeln 90 Grad sowie 270 Grad ein Minimum, beziehungsweise ein Maximum aufweist und damit die "Messstelle"für die Nickbewegung darstellt. Abschließend werden bei jedem Schleifendurchlauf die Variablen für *leftBlade* und *rightBlade* aktualisiert, falls der derzeitige Wert für die *contribution* ein neues Minimum oder Maximum darstellt. Zum Ende werden die Indizes für die Rotorblätter zurückgeliefert, die dann für die Berechnung des Drehmomentes in Nickrichtung benötigt werden. Dabei werden diese zusätzlich mit dem Abstand zur "Messstelle"gewichtet werden, da ein voll ausgeprägter Beitrag nur dann vorhanden ist, wenn das Rotorblatt sich in etwa bei 90 oder 270 Grad befindet. Zwischen 45 und 90 Grad, sowie 225 und 270 Grad findet die oben genannte Gewichtung statt.

Die Ermittelung der Rotorblätter für die Roll-Bewegung erfolgt analog zu Algorithmus 1. Einziger Unterschied besteht darin, dass der Wert für *contrib* in Zeile sechs nicht mit dem Sinus sondern mit dem Kosinus des Azimutwinkels berechnet wird. Auch hier wieder analog zum obigen Verfahren deshalb, da der Kosinus bei den Azimutwinkeln 0 und 180 Grad ein Maximum und Minimum besitzt.

5.3.4 Tail Rotor

Hauptaufgabe der Klasse ist es, den aktuellen Zustand des Heckrotors zu kapseln und den Schub um die Z-Achse zu berechnen. Als Eingabe bekommt der Heckrotor lediglich ein Flag für den Status, ob die Triebwerke gestartet sind oder nicht, sowie den `Yaw`-Input von der Steuerung. Da der Heckrotor wie oben beschrieben über keinerlei zyklische Blattverstellung verfügt, vereinfacht sich die Berechnung des Schubs erheblich. Dies geschieht wie aus Listing 9 ersichtlich in der Methode `CalculateLift`, die analog zur Berechnung aus Listing 8 erfolgt. Einziger Unterschied an dieser Stelle ist, dass der Einstellwinkel nicht umgerechnet wird, da keine zyklische Steuerung stattfindet.

```
1  float tailRotorLift = TailRotor->CalculateLift();
2  FVector torque = FVector(0.f, 1.f, 0.f) * tailRotorLift *
     OffsetTailRotor;
3
4  ...
5
6  L = torque * DeltaTime;
7  L *= 0.88; //some linear damping
```

Listing 9: Berechnung Heckrotor

Damit ist diese nur von der Umdrehungszahl und dem kollektiven Blattanstellwinkel abhängig. In Kombination mit einem Offset-Vektor kann die übergeordnete MovementCompnent-Klasse das Drehmoment berechnen. Hier an dieser Stelle mit `L` bezeichnet. Normalerweise würde der Heckrotor durch seine Widerstandskraft auch noch ein Roll-Moment erzeugen. Dies ist üblicherweise vom Piloten auszugleichen und wird an dieser Stelle vernachlässigt.

5.4 Translation

Wie im Abschnitt 4.2.2 beschrieben, verwendet die Simulation das Semi-implizite Euler-Verfahren. Das bedeutet, es wird zuerst die Geschwindigkeit für den kommenden Zeitschritt aktualisiert und danach die davon abhängige Translation. Für die Translation spielt auch die Orientierung der Rotorscheibe eine wichtige Rolle. Anhand des Neigungswinkels zwischen der globalen Y-Achse und der X-Achse in Rotorkopf-Koordinaten kann der Schub in Teilkräfte zerlegt und sein Beitrag zur Verschiebung des Hubschraubers in X-Richtung ermittelt werden. Damit ist eine Bewegungsrichtung der Translation berechnet. Für die Translation in Y-Richtung wird analog der Neigungswinkel zu den Seiten (Steuerbord und Backbord) mit der globalen Y-Achse berechnet. Dabei sei gesagt, dass der Einfachheit halber die Rotationswinkel um die Roll-, Nick- und Gierachse auf jeweils insgesamt 180 Grad beschränkt ist. Das dient dem Zweck, dass sich

der Beitrag für die Schubkomponente in Y-Richtung (um Höhe zu gewinnen) leichter berechnen lässt. Er beträgt dann nämlich einfach dem Gegenwinkel. Auf diese Art und Weise sind alle Freiheitsgrade der Translation bestimmt und über die numerische Integration wird die Geschwindigkeit mittels der Beschleunigung berechnet. Zum Abschluss wird dann die `Position`-Variable des Pawn-Objektes gesetzt. Die Unreal Engine stellte der `MovementComponent`-Klasse zwar auch eine `Velocity`-Variable zur Verfügung, allerdings zeigt sich selbst beim Aufrufen der Funktion `UpdateComponentVelocity` keine Änderung des Bewegungszustandes des Pawns. Möglicherweise ist dies auch die bessere Wahl, da vermutlich bei erfolgreichem Update der Geschwindigkeit mit den „Bord-Mitteln" auch die Bewegungsgleichungen mit der Physx-Engine gelöst würden. Dies galt es ja weitestgehend zu vermeiden.

5.5 Rotation

In diesem Abschnitt wird die Berechnung der neuen Orientierung ausgehend von dem Code aus Listing 9 betrachtet. Der Heckrotor erzeugt ein Drehmomemt um die Hochachse, welches im `Torque`-Akkumulator hinzugefügt wird. Die aktuelle Winkelgeschwindigkeit und folglich die neue Orientierung `q` wird in Listing 10 berechnet.

```
1  q = UpdatedComponent->GetComponentRotation().Quaternion();
2
3  FQuatRotationMatrix quatRotationMatrix = FQuatRotationMatrix(q);
4  FMatrix rotMatrix = quatRotationMatrix.Make(quatRotationMatrix.
     ToQuat());
5
6  w = rotMatrix.TransformVector(jBodyInv.TransformVector(rotMatrix
     .GetTransposed().TransformVector(L)));
7
8  FQuat QuatAngularVelocity = w.ToOrientationQuat() * 0.5f;
9
10 QuatAngularVelocity.Normalize();
11 q = QuatAngularVelocity * q;
```

Listing 10: Integration der neuen Orientierung

Die Winkelgeschwindigkeit für die Gierachse ist hauptsächlich vom Drehmoment durch den Heckrotor abhängig. Wie dieses berechnet wird, wurde oben schon dargestellt. Die weiteren Komponenten, die die Winkelgeschwindigkeit um die Nick- und Rollachse beeinflussen sind die Hauptrotormomente. Da diese durch die unsymmetrische Auftriebsverteilung am Hauptrotor entstehen, muss zuerst geschaut werden, wie der Azimutwinkel des Hauptrotors ist. In Abhängigkeit davon können dann wie im Algorithmus 1 die Drehmomente für die Nick- und Rollachse bestimmt und schließlich die Drehimpulse berechnet werden und zu `L` akkumuliert werden.

5.6 Aktualisierung der Simulation

Innerhalb der Funktion *TickComponent(...)* wird zum Schluss des Simulationsdurchlaufes die Funktion *SafeMoveUpdatedComponent(Position, Rotation, true, Hit)* aufgerufen. Aus der Methodensignatur ist abzulesen, dass die ersten beiden Parameter die neue Position und Rotation sind. Die anderen Parameter beziehen sich auf den Sweep-Test für die Kollisionserkennung, sowie das entsprechende *Hit*-Struct, welches im Falle der Kollision, die notwendigen Daten beinhaltet. Zum Abschluss wird die Rotationsmatrix aus dem aktuellen Orientierungs-Quaternion bestimmt, um eine aufwändige Orthogonalisierung zu sparen. Danach endet die Arbeit der *MovementComponent* und wartet darauf, dass das nächste Tick-Event von der übergeordneten Pawn-Instanz kommt, worauf ein neuer Simulationsschritt angestoßen wird. Der Aufruf der *TickComponent*-Methode ist nicht fest an das Rendering der Simulation gebunden, sodass davon ausgegangen werden kann, dass pro Renderdurchlauf, die Simulationsmethoden mehrfach aufgerufen werden können.

6 Fazit

Wie in Abschnitt 4.1 dargestellt ist, wurden eine Reihe von Idealisierungen angenommen, um zum einen, die Übersichtlichkeit zu gewährleisten und das Gesamtsystem nicht zu komplex zu gestalten. Zum anderen, ist dadurch eine interaktive Simulation sichergestellt. Mitnichten lässt sich sagen, dass die im Rahmen dieser Arbeit erstellte Simulation fernab von dem ist, was echte Trainingssimulatoren zu bieten haben. Sowohl hinsichtlich Realismus als auch Komplexität der physikalischen Systeme. Nichtsdestotrotz bekommt man auch mit diesem kleinen Simulatorprojekt ein Gefühl dafür, wie sich verschiedene aerodynamische Effekte auswirken und sukzessive im Aufbau der Arbeit zu einem Gesamtsystem integriert werden. Als hinderlich hat sich die mangelnde Verfügbarkeit von aktuellen Leistungsdaten herausgestellt. Verständlicherweise sind aktuelle Leistungsparameter ein gut Gehütetes Geschäftsgeheimnis, allerdings sind auch so trivial anmutende Daten, wie beispielsweise die Leistungspolare nicht ohne Weiteres öffentlich verfügbar. So ist zwar über Umwege und sehr viel kleinteiliger Suche bekannt, welche Rotorblattbezeichnung die Richtige ist für die EC 135, allerdings lassen sich in den Datenbanken keinerlei Daten finden. An dieser Stelle wurde die Rotorcharakteristik parametrisiert, dass verschiedenste Profildaten eingelesen werden können. Dies ist mittels des einfachen CSV-Import in die Engine auch kein großes Problem, sodass eine Vielzahl von Profildaten, insbesondere den NACA Profilen, ausgetestet werden kann. Ebenfalls war die Ermittlung des Trägheistensors ein Kompromiss. Zwar liegt ein gut entwickeltes 3D-Model des Hubschraubers vor, allerdings lassen sich damit nur sehr schwer die Massen der einzelnen Bauteile abschätzen und zu einem Trägheitstensor zusammenfassen. Zudem stellt die Bestimmung von Trägheitstensoren auf komplexen konvexen Polyedern laut [Eberly, 2015] auch eine nicht zu unterschätzende Aufgabe dar. Daher wurde hier eine Bounding-Box mit konstanter Massenverteilung angenommen, die zumindest eine Abschätzung nach oben darstellt, da der eigentliche Rumpf wesentlich kompakter gebaut ist und damit Drehkräfte einen unmittelbar höheren Einfluss auf die Rotation haben. Ein Verzicht auf die „Bord-Mitteln“ der Unreal Engine zeigte, wie gut die PhysX-Engine integriert ist. An einigen Stellen musste explizit drum herum gearbeitet werden. Die Verwendung einer eigenen MovementComponent-Klasse stellt dahingehend unter der Vielzahl an möglichen Lösungsansätzen, die Vernünftigste dar. Sicherlich lässt sich der Simulations-Code noch weiter und besser Modularisieren, auch im Sinne einer Verwendung innerhalb der Engine als eigentständiges Hubschrauber-Plugin. Darauf wurde aber bewusst zum Zweck der besseren Illustration verzichtet. Ärgerlich sind ein paar Mehrdeutigkeiten in der Dokumentation der Unreal Engine. So beinhaltet die MovementComponent-Klasse ein Feld für den PawnOwner, der der MovementComponent übergeordnet ist. Dieses Feld ver-

ursachte stets unnachvollziehbare Laufzeitfehler, obwohl beim Zugriff explizit auf die von der Engine bereitgestellten Cast-Funktionen zurückgegriffen wurde und Fehler in Bezug auf Null-Pointer und dergleichen ausgeschlossen waren. Dies bedingte den Umstand, die MovementComponent zu Beginn schon mit allen irgendwann benötigten Daten zu initialisieren, statt das diese sich die Daten bei Bedarf über eine entsprechende Getter-Funktion vom Pawn-Objekt holt. Für eine Weiterentwicklung ist es vorstellbar, die PhysX-Engine vollumfänglich zu integrieren, sodass eine Reihe von Interfaces innerhalb der Unreal Enine besser genutzt werden kann und weniger Boilerplate-Code notwendig ist. Sehr positiv sind eine Reihe von Helfer-Klassen oder ähnlicher Funktionalitäten aufgefallen, wie der automatischen Interpolation der Auftriebs- oder Widerstandsbeiwerte. Auch funktionierte die Steuerung mittels Joysticks und Weitergabe der entsprechenden Events nach anfänglichen Konfigurationsproblemen tadellos. Abschließend ist zu sagen, dass die hier aufgebaute Simulation einen Grundstein für eine Weiterentwicklung eines Hubschraubersimulators darstellt. Aufgrund der Vereinfachungen ist es ein Starrkörper-Simulator mit einer Hubschraubersteuerung. Allerdings wird schnell klar, dass selbst in dieser einfachen Konfiguration die Steuerung eines Hubschraubers eine überaus anspruchsvolle Aufgabe ist. Dies geschieht alles im Editor und sowohl die Rotordaten, als auch die 3D-Meshes für Rumpf und Rotoren können interaktiv geändert werden, ohne das ein neukompilieren der Anwendung notwendig ist. Dies beschränkt sich jedoch nur auf die Simulation mittels Editor. Wird die Simulation als fertig gebautes Projekt exportiert, entfällt der Editor und die eingestellten Parameter fallen auf die Default-Parameter der Implementation zurück.

7 Ausblick

Für die Weiterentwicklung des Simulators gibt es eine Reihe von Ideen und Verbesserungen, die sich an dieser Stelle anbieten. Andere Flugdynamik-Systeme wie zum Beispiel YASim[5] oder JSBSim[6] simulieren den Antriebsstrang in unterschiedlicher Granularität mit. Das hat zum einen den Vorteil, dass Triebwerksparameter dynamisch im Cockpit visualisiert werden können, Start-Prozeduren über den schlichten Start-/Stop-Zustandswechsel hinausgehen und weitere Kraftauswirkungen während der Simulation berechnet werden können. Zudem sind verschiedene Lastszenarios denkbar, um auch die Grenzen beziehungsweise die Überlastung eines Flugmusters ausserhalb des vom Hersteller spezifizierten Einsatzbereichs zu zeigen. Erweiterte Trainigszenarios wie die Autorotation und im speziellen die Autorotationslandung, also der Flugzustand, in dem der Rotor sich durch die Anströmung von unten selbst in Rotation hält, bedürfen auch einer Simulation des Antriebsstranges, da hier Reibungsverluste im Triebwerk und den Getrieben simuliert werden müssen, um eine realistische Fluglage zu beschreiben. Die in die Unreal Engine integrierte PhysX-Engine bietet im Ursprungszustand nur rudimentäre Unterstützung für Triebwerke. Es lassen sich allenfalls Verbrennungsmotoren definieren, da für Fahrzeuge schon die entsprechenden Funktionalitäten implementiert sind, die Motoren, Getriebe und Aufhängungen umsetzen. Eine weitere Verbesserung hinsichtlich der Simulationsqualität stellt die zu Beginn der Arbeit erwähnte Computational Fluid Dynamics (CFD) dar. Nicolas Klee zeigte in seiner Arbeit [Klee, 2018] einen Ansatz dafür, wie sich eine Windsimulation effizient innerhalb einer Spieleumgebung umsetzen lässt. Dabei erfüllte er die obere Schranke von einer Millisekunde Berechnungszeit zuverlässig und kann als Grundlage für eine fortgeschrittene Hubschraubersimulation genommen werden. Möglich wäre es, wie in der Arbeit gezeigt, eine Simulationsdomäne für die CFD an den Hubschrauber zu heften und dementsprechend zu diskretisieren, dass im Nahfeld eine Vielzalh von Effekten simuliert werden kann, fernab dessen allerdings die Auflösung der Windsimulation allmählich abnimmt, um die Performanz zu steigern. Es ist natürlich zu Prüfen, inwiefern die Ergebnisse im Ansatz von Klee hauptsächlich visuell stimmig sind oder aber auch den komplexen numerischen Anforderungen einer CFD hinreichend genügen. Ein wichtiges Trainingsszenario, gerade für Piloten mit einer hohen Zahl an Aussenlandungen, sprich Landungen auf unbefestigten Landeplätzen oder ähnlichem, ist eine Flugsituation, die sich Brown-Out nennt. Der Pilot ist oftmals unter Sichtflugbedingungen unterwegs und hat daher die Anforderung an markante Geländemarken, um die aktuelle Fluglage erfolgreich

[5] `wiki.flightgear.org/YASim`

[6] `http://jsbsim.sourceforge.net/`

beurteilen zu können und situationsgerecht zu agieren. Gerade in der Rettungsfliegerei ist es häufig der Fall, dass im Landeanflug lose Partikel durch den Rotorabwind aufgewirbelt werden und so der oben genannte Brown-out entsteht, sodass die Sicht dramatisch schlechter wird. Friedmann gibt in [Friedmann, 2016] Flugsituationen mit Brown-out als eine signifikante Ursache an in Bezug auf ernsthafte Flugunfälle. Dementsprechend ist es wünschenswert solche Situationen in einen Simulator zu integrieren, damit sie dort gefahrlos trainiert werden können. Die Firma JangaFX[7] bietet eine Software mit dem Namen VectorRayGen an. Dabei handelt es sich um einen Node-basierten Vektorfeld-Editor. Damit lassen sich innerhalb der Software Vektorfelder unterschiedlichster Auflösung definieren und dann in die Unreal Engine importieren. Dabei wird das erzeugte Vektorfeld als Modifier für die Unreal-internen Partikeleffekte verwendet. Auf diese Weise lassen sich visuell sehr hochwertige und glaubwürdige Rauch- und Staubeffekte darstellen. Kombiniert man ein solches Partikelsystem mit dem von Unreal integriertem Line-Tracing, ließe sich so in Abhängigkeit der Flughöhe und des Untergrundes ein Brown-out simulieren. Weiterhin ist zu prüfen, ob ein Terrain-Mesh, dass mit der Unreal Engine erstellt wurde, sich annotieren lässt, sodass die Beschaffenheit verschiedener Untergründe umgesetzt werden kann. Ebenfalls ein wichtiger Leistungsparameter im Hubschrauberflug stellt der Bodeneffekt dar. Grob vereinfacht erzeugt der Rotorabwind in Bodennähe ein Luftkissen, wodurch der Auftrieb vergrößert und der Leistungsbedarf des Hubschraubers innerhalb des Bodeneffektes verringert wird. An dieser Stelle wären die Strahltheorie in Kombination mit einer CFD zu kombinieren. Aber auch ein naiver Ansatz mittels Line-Tracing und einer entsprechend zurecht geschnittenen Leistungskurve könnte den Bodeneffekt in erster Instanz glaubwürdig approximieren. Um gerade Landemanöver, auch in Extremsituationen, bewerten zu können ist eine ausgeklügelte Kollisionserkennung und Kollisionsbehandlung vonnöten. Gerade das fragil anmutende Landegestell ist bei unsanften Landungen enormen Kräften ausgesetzt. Aber auch Schwingungen oder Momente können sich auf die gesamte Rumpfintegrität, insbesondere des Heckauslegers, auswirken. Um den Realismus des Renderings noch weiter zu steigern wäre eine Integration von echten Geländedaten denkbar. Mike Fricker, Technikchef bei Epic, veröffentlichte ein Plugin[8] für die Unreal Engine mit dem es möglich ist, Kartenausschnitte aus OpenStreet-Map zu exportieren und daraus Terrain-Meshes zu erzeugen. Teilweise, je nach Datenlage, beinhalten die so erzeugten Terraindaten auch Gebäudestrukturen und sorgen so für eine leichtere Orientierung des Piloten während des Flugs. Zu guter Letzt sei der Ansatz einer VR-Anwendung erwähnt, der sicherlich in der Lage ist, den Realismus, zumindest subjektiv zu steigern.

[7] `https://jangafx.com/`

[8] `https://github.com/ue4plugins/StreetMap`

Eine Integration in die Unreal Engine stellt keinen allzu großen Aufwand dar, weil entsprechende Interfaces und Wrapper-Klassen bereits fertig zur Verfügung stehen. Ein Grund für die Verwendung von Full-Flight Simulatoren mit sechs Freiheitsgraden ist der, dass Piloten zu einem gewissen Teil Fluglagen über die Kräfte, die auf ihren Körper wirken, abschätzen und beurteilen. Eine VR-Anwendung könnte, selbst beim nicht Vorhandensein einer kinematischen Plattform, in der Lage sein, diese Kräfte zu suggerieren. Natürlich ist höchste Vorsicht geboten, um Übelkeit bei den Anwendern zu vermeiden, allerdings hat die physikalische Simulation als solches darauf weniger Einfluss als das Rendering.

Literatur

[Gar, 2012] (2012). *Hubschrauber: Vom Pitcairn Autogiro zum Sikorsky S-97 Raider*. Garant, Renningen.

[Abarr, 2018] Abarr, C. (2018). H135 - quigdao united. https://airbus-h.assetsadobe2.com/is/image/content/dam/products-and-solutions/commercial-helicopters/h135/EXPH-0442-56R.jpg?wid=991&fit=fit,1&qlt=85,0.

[Abbott and Von Doenhoff, 1959] Abbott, I. H. and Von Doenhoff, A. E. (1959). *Theory of wing sections, including a summary of airfoil data*. Courier Corporation.

[Airbus Helicopters, 2014] Airbus Helicopters (2014). Characteristics_ec135 t3e/p3e - airbus helicopters. https://web.archive.org/web/20141014075448/http://www.airbushelicopters.com/site/en/ref/Characteristics_1349.html.

[Bittner, 2014] Bittner, W. (2014). *Flugmechanik der Hubschrauber*. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Dahmen and Reusken, 2008] Dahmen, W. and Reusken, A. (2008). *Numerik für Ingenieure und Naturwissenschaftler*. Springer-Lehrbuch. Springer, Berlin, 2., korrigierte aufl. edition.

[Eberly, 2015] Eberly, D. H. (2015). *Game physics*. CRC Press, Boca Raton, second edition edition.

[Friedmann, 2016] Friedmann, L. M. (07.11.2016). *Echtzeit-Simulation des Rotorabwinds von Hubschraubern und Kipprotor-Flugzeugen mit der Lattice-Boltzmann Methode*. Dissertation, Technische Universität München, München.

[Froude, 1878] Froude, W. (1878). *On the Elementary Relation between Pitch, Slip and Propulsive Efficiency*, volume 19. Transactions of the Institution of Naval Architects.

[G. Polz, 1987] G. Polz, D. S. (1987). New aerodynamic rotor blade design at mbb.

[Grahamuk, 2012] Grahamuk (2012). Auftriebsbeiwert. File: Auftriebsbeiwert.svg.

[Johnson, 1994] Johnson, W. (1994). *Helicopter theory*. Dover and London : Constable, New York.

[Kampa et al., 1997] Kampa, K., Enenkl, B., Polz, G., and Roth, G. (1997). Aeromechanic aspects in the design of the ec 135.

[Klee, 2018] Klee, N. (2018). Scalable wind simulation for landscape rendering. Master's thesis, Universität Koblenz-Landau.

[Leishman, 2006] Leishman, J. G. (2006). *Principles of helicopter aerodynamics*, volume 18 [i.e. 12] of *Cambridge aerospace series*. Cambridge University Press, Cambridge, 2nd ed. edition.

[Polte, 2011] Polte, H.-J. (2011). *Hubschrauber: Geschichte, Technik und Einsatz*. Mittler, Hamburg, 5., völlig neu überarb. und erw. aufl. edition.

[Rankine, 1865] Rankine, W. (1865). *On the Mechanical Principles ofthe Action of Propellers*, volume 4. Transactions of the Institution of Naval Architects.

[Surek and Stempin, 2007] Surek, D. and Stempin, S. (2007). *Angewandte Strömungsmechanik: Für Praxis und Studium*. B.G. Teubner Verlag / GWV Fachverlage GmbH Wiesbaden, Wiesbaden.

[Tipler et al., 2015] Tipler, P. A., Mosca, G., and Wagner, J. (2015). *Physik: Für Wissenschaftler und Ingenieure*. Springer Berlin Heidelberg, Berlin and Heidelberg, 7. aufl. 2015 edition.

[Wall, 2015] Wall, B. G. (2015). *Grundlagen der Hubschrauber-Aerodynamik*. Springer Berlin Heidelberg, Berlin, Heidelberg.