Ufo-Handbuch

Version 3-2-9q

Prof. Dr. Robert Gold

Technische Hochschule Ingolstadt
Wintersemester 2025/26

Das Python-Modul ufosim3_2_9q.py mit den Klassen UfoSim und UfoPView simuliert ein kleines Fluggerät, im Folgenden Ufo genannt. Verwendet werden kann ein Ufo beispielsweise, um Waren in die nähere Umgebung zu liefern. Das Ufo startet dabei in einem fest definierten Punkt. Typischerweise fliegt es verschiedene Zielpunkte an, um danach zum Ausgangspunkt zurückzukehren.

Der Name Ufo steht für Unified Flying Object, weil damit verschiedene Fluggeräte wie Drohnen, Hubschrauber, Ballone, Raumschiffe der Außerirdischen, etc. simuliert werden können.

1 Flugdaten

Die Position des Ufos wird mit kartesischen Koordinaten (x, y, z) in einem dreidimensionalen Koordinatensystem dargestellt. Der Ursprung des Koordinatensystems ist die anfängliche Startposition des Ufos. Die x-y-Ebene ist die Erdoberfläche. (Wie wir wissen, ist die Erde flach wie eine Scheibe?) Das Ufo selbst ist in der Simulation idealisiert punktförmig.

Sie können sich vorstellen, dass die x-Achse die West-Ost-Richtung und die y-Achse die Süd-Nord-Richtung ist (Abb. 1). Die z-Koordinate ist die Flughöhe (Abb. 2). Eine negative Flughöhe bedeutet einen Absturz des Ufos.

Das Ufo hat außerdem eine Flugrichtung (d, i) und eine Fluggeschwindigkeit v längs der Flugrichtung ($0 \le v \le$ VMAX). Die Richtung d ist der Winkel in Grad zwischen der positiven x-Achse und der Projektion des Flugrichtungsvektors auf die x-y-Ebene ($0 \le d \le 359$). Die Neigung i ist der Winkel in Grad zwischen der positiven x-Achse und der Projektion des Flugrichtungsvektor auf die x-z-Ebene ($-90 \le i \le 90$). Eine negative Richtung zeigt nach unten (Abb. 3).

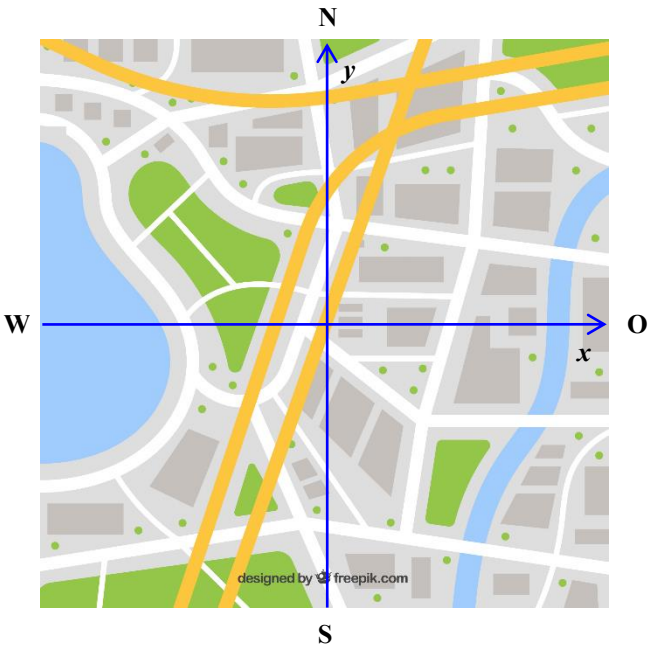

Abb. 1: *x*-*y*-Ebene

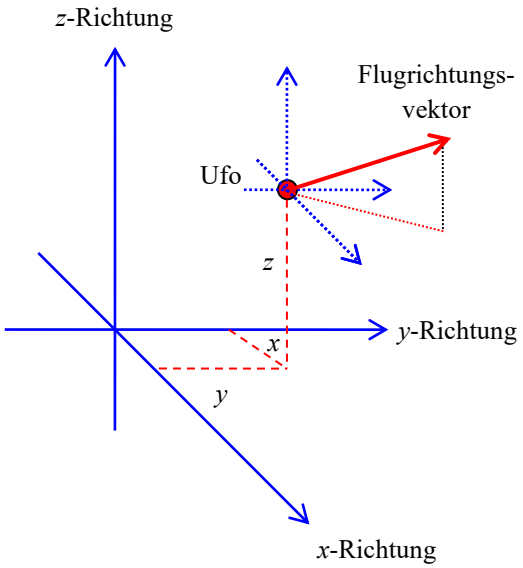

Abb. 2: Dreidimensionale Ansicht

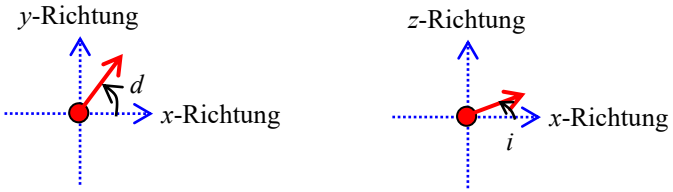

Abb. 3: Projektion auf die *x*-*y*-Ebene bzw. auf die *x*-*z*-Ebene

Beispiele:

$x = 75$, $y = 100$, $z = 197$, $d = 60$, $i = 22$: entspricht ungefähr der Abbildung

$x = 0$, $y = 0$, $z = 0$, $d = 90$, $i = 90$: Startposition im Ursprung, Richtung senkrecht nach oben

$d = 180$, $i = 0$: Das Ufo fliegt in parallel zur x-Achse in negativer Richtung.

$i = 90$: Das Ufo steigt senkrecht auf.

$i = -90$: Das Ufo fliegt senkrecht nach unten.

$z \le 0$, $v > 1$: Das Ufo stürzt ab.

$z \le 0$, $v = 1$: Das Ufo landet. Man darf also nur mit Geschwindigkeit 1 landen.

Verringerung von d: Das Ufo fliegt nach rechts.

Verringerung von i: Das Ufo neigt sich nach unten.

Das Ufo hat also folgende Flugparameter:

Name	Bedeutung	Datentyp	Einheit	Bedingung	Initialwert
x	x-Koordinate	Gleitpunktzahl	m		0
y	y-Koordinate	Gleitpunktzahl	m		0
z	z-Koordinate	Gleitpunktzahl	m		0
v	Geschwindigkeit	Ganze Zahl	km/h	$0 \le v \le$ VMAX	0
d	Richtung	Ganze Zahl	Grad	$0 \le d \le 359$	90
i	Neigung	Ganze Zahl	Grad	$-90 \le i \le 90$	90

Leider wird das Ufo nur simuliert. Der Vorteil davon ist, dass uns das Gerät nicht auf dem Kopf fallen kann. Außerdem haben wir immer schönes Wetter. Wind und Regen haben keinen Einfluss.

VMAX ist zurzeit auf 15 km/h gesetzt. Man kann also recht flott Bahnen über den Himmel ziehen. (Naja, für Außerirdische wäre es wohl eher Schneckentempo.)

Das Ufo misst die seit dem letzten Reset geflogene Distanz (*dist*) längs der Flugrichtung und die dafür benötigte Zeit (*ftime*). Wenn die Höhe kleiner oder gleich 0 ist, läuft die Zeit nicht weiter.

Weitere Flugdaten:

Name	Bedeutung	Datentyp	Einheit	Bedingung	Initialwert
dist	Geflogene Strecke	Gleitpunktzahl	m	$dist \ge 0$	0
ftime	Flugzeit	Gleitpunktzahl	s	$ftime \ge 0$	0

2 Auslesen der Flugdaten

Die Flugdaten können durch ein Programm nicht direkt, sondern nur über die folgenden Funktionen ausgelesen werden:

Name	Bedeutung der Funktion	Parameter	Rückgabetyp
get_x	Holen der *x*-Koordinate	keiner	Gleitpunktzahl
get_y	Holen der *y*-Koordinate	keiner	Gleitpunktzahl
get_z	Holen der *z*-Koordinate	keiner	Gleitpunktzahl
get_v	Holen der Geschwindigkeit	keiner	Ganze Zahl
get_d	Holen der Richtung	keiner	Ganze Zahl
get_i	Holen der Neigung	keiner	Ganze Zahl
get_dist	Holen der geflogenen Strecke	keiner	Gleitpunktzahl
get_ftime	Holen der Flugzeit	keiner	Gleitpunktzahl

Diese Funktionen werden von der Simulationsklasse bereitgestellt und können verwendet werden.

3 Steuerung des Ufos

Das Ufo lässt sich längs des Flugvektors beschleunigen. Es nur eine konstante Beschleunigung mit dem Beschleunigungswert ACCELERATION möglich. Die Konstante hat den Wert 1 km/h/0.1s, d.h. in einer Sekunde erhöht sich die Geschwindigkeit um 10 km/h. Die Verzögerung geschieht analog mit –ACCELERATION km/h/0.1s.

Da sich das Ufo immer längs des Flugrichtungsvektors bewegt, müssen zum Steuern des Ufos die Richtung und die Neigung verändert werden.

Die Simulationsklasse bietet zur Steuerung folgende Funktionen an.

Name	Bedeutung der Funktion	Parameter	Rückgabetyp
request_delta_v	Setzen eines Wunsches zur Geschwindigkeitsänderung. Das Ufo beschleunigt bzw. verzögert selbstständig auf den neuen Geschwindigkeitswert. Die Bedingung $0 \leq v \leq$ VMAX wird vom Ufo eingehalten. Beschleunigungs-/Verzögerungswert: ACCELERATION bzw. –ACCELERATION	Ganze Zahl	keiner
request_delta_d	Setzen eines Wunsches zur Richtungsänderung. Das Ufo dreht sich selbstständig in die neue Richtung. Die Bedingung $0 \leq d \leq 359$ wird vom Ufo eingehalten. Bei einer Linksdrehung über 359 Grad, dreht das Ufo	Ganze Zahl	keiner

	weiter auf 0 Grad, 1 Grad, … Entsprechendes gilt bei einer Rechtsdrehung. Die Drehung erfolgt mit 10 Grad / s.		
request_delta_i	Setzen eines Wunsches zur Neigungsänderung. Das Ufo neigt sich selbstständig in die neue Richtung. Die Bedingung $-90 \le i \le 90$ wird vom Ufo eingehalten. Die Neigung erfolgt mit 10 Grad / s.	Ganze Zahl	keiner
set_d	Setzen der neuen Richtung. Die Richtung wird sofort geändert. Dies ist nicht wirklich realistisch, aber praktisch. Falls der Parameter < 0 oder > 359 ist, erfolgt keine Änderung.	Ganze Zahl	keiner
set_i	Setzen der neuen Neigung. Die Neigung wird sofort geändert. Dies ist nicht wirklich realistisch, aber praktisch. Falls der Parameter < –90 oder > 90 ist, erfolgt keine Änderung.	Ganze Zahl	keiner

Es ist zu beachten, dass das Ausführen von Änderungswünschen Zeit braucht, z.B. Ändern der Geschwindigkeit von 0 auf 15 km/h: 1.5 s, Drehen um 180 Grad: 18 s. Es gibt keine verzögerungsfreie Funktion set_v. Das wäre zu weit weg von der Realität. Die Verwendung von set_d und set_i erleichtert die Programmierung sehr.

Beispiele:

vorher	Funktionsaufruf	nachher
$v = 10$	request_delta_v(4)	Nach 0.4s ist $v = 14$
$v = 10$	request_delta_v(–20)	Nach 1s ist $v = 0$
$i = 45$	request_delta_i(–65)	Nach 6,5s ist $i = -20$
$d = 45$	request_delta_d(330)	Nach 33s ist $d = 15$
$i = 45$	set_i(–20)	Nach 0s ist $i = -20$
$i = 45$	set_i(100)	Keine Änderung

Das Ufo regelt seine Antriebe selbstständig so, dass die eingestellte Geschwindigkeit, Richtung und Neigung genau eingehalten werden.

Zum Erstellen, Starten und Beenden der Simulation sind noch die folgenden Funktionen wichtig.

Name	Bedeutung der Funktion	Parameter	Rückgabetyp
UfoSim	Erstellt ein Simulationsobjekt. Der Rückgabewert muss einer Variablen, z.B. sim, zugewiesen werden damit das Simulationsobjekt weiter verwendet werden kann.	keiner	Simulations-objekt
start	Startet die Simulation. Der erste Parameter ist der Speedup. Normalerweise läuft die Simulation in Echtzeit, d.h. eine Sekunde in der Simulation entspricht einer Sekunde in der Realität. Wem das zu langsam ist, kann den Speedup auf einen ganzzahligen Wert zwischen 1 und 25 setzen. Speedup 1 bedeutet Simulation in Echtzeit. Der zweite Parameter ist die Skalierung der View. Erlaubt sind ganzzahlige Werte zwischen 1 und 100. Die Skalierung wird später besprochen. Der dritte Parameter ist eine Liste von Zielen. Beispielsweise bedeutet [(20.0, 20.0)], dass es ein Ziel mit den Koordinaten 20.0, 20.0 gibt. Die Zieleliste dient nicht zur Steuerung des Ufos, sondern ist nur zur Anzeige der Ziele in der View erforderlich (siehe unten).	Zwei ganze Zahlen und eine Liste von Tupeln von Gleit-punkt-zahlen	keiner
terminate	Beendet die Simulation	keiner	keiner

4 Programmierbeispiel

Zum besseren Verständnis der Funktionen der Simulation folgt ein kommentiertes Programmierbeispiel in Python. Das Beispiel fliegt das Ufo von (0.0, 0.0, 0.0) nach (20.0, 20.0, 0.0) (Abb. 5).

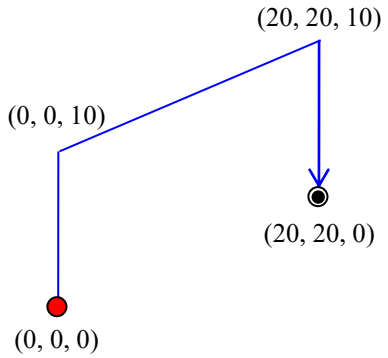

Abb. 5: Beispielflug

```
from ufosim3_2_9q import UfoSim
import math
import time

# The simulation is created and started with a speedup, a view with scaling 10
# and the target (20.0, 20.0).
sim = UfoSim()
speedup = 5
scaling = 10
destinations = [(20.0, 20.0)]
sim.start(speedup, scaling, destinations)

# The following code flies the Ufo to position (20, 20, 0).

# The ufo flies vertically upwards at 10 km/h.
print("takeoff with 10 km/h to alt 10 m...")
sim.set_i(90)
sim.request_delta_v(10)

# Before reaching altitude 8m, the ufo brakes to 1 km/h.
# The loop waits briefly before the loop condition is checked again.
# For greater speedup, the wait time is shorter.
while sim.get_z() < 8:
    time.sleep(0.1/speedup)
print("...slow down to 1 km/h... ")
sim.request_delta_v(-9)

# When the ufo reches altitude 9.95m, it stops and aligns itself horizontally.
while sim.get_z() < 9.95:
    time.sleep(0.1/speedup)
print("...stop and turn horizontal")
sim.request_delta_v(-1)
sim.set_i(0)

# We continue toward 45 degrees.
# The distance to be flown is the square root of 20 * 20 + 20 * 20.
sim.set_d(45)
dist = sim.get_dist() + math.sqrt(20 * 20 + 20 * 20)

# The ufo accelerates to 15 km/h.
print("go " + str(45) + " deg with 15 km/h...")
sim.request_delta_v(15)

# When the vertical distance to the destination is 4 m, the ufo brakes to 1 km/h.
while dist - sim.get_dist() > 4:
    time.sleep(0.1/speedup)
print("...slow down to 1 km/h... ")
```

```
sim.request_delta_v(-14)

# When the vertical distance to the destination is 0.05 m, the ufo stops.
while dist - sim.get_dist() > 0.05:
   time.sleep(0.1/speedup)
print("...stop")
sim.request_delta_v(-1)

# The ufo flies vertically downwards at 10 km/h.
print("landing with 10 km/h")
sim.set_i(-90)
sim.request_delta_v(10)

# When the altitude reaches 3 m, the ufo brakes to 1 km/h.
while sim.get_z() > 3:
   time.sleep(0.1/speedup)
print("...slow down to 1 km/h...")
sim.request_delta_v(-9)

# The ufo has landed when the altitude is less than or equal to 0.
while sim.get_z() > 0:
   time.sleep(0.1/speedup)
print("...happily landed")

# Terminate the simulation
sim.terminate()
```

Beim Ausführen müssen sich das Beispielskript, die Simulation ufosim3_2_9q.py und die benötigten Bilddateien 114060-OOYHOE-336.png, thi_icon_258.png, ufo_icon2.png im selben Verzeichnis befinden.

Zu beachten ist, dass die Funktionen requestDeltaV, requestDeltaD, requestDeltaI, setD und setI beim Aufruf ein ganzzahliges Argument verlangen. Ein Aufruf mit einer Gleitpunktzahl kann einen Laufzeitfehler ergeben.

In den Warteschleifen wird die Ausführung des Programms jeweils 0.1 /speedup Sekunden (= 0.1/5 s = 20 ms) durch Aufruf von time.sleep(0.1/speedup) pausiert. Die Pausierung kann auch komplett herausgenommen werden, z.B. durch Ersetzen von time.sleep(0.1/speedup) durch pass.

5 View

Ein echtes Fluggerät kann während Testflügen beobachtet werden. Das geht in der Simulation leider nicht. Um eine kontinuierliche Beobachtung zu ermöglichen, hat der Ersteller der Simulation eine Klasse UfoPView in die Simulation eingebaut. Die View zeigt verschiedene Daten an (Abb. 6).

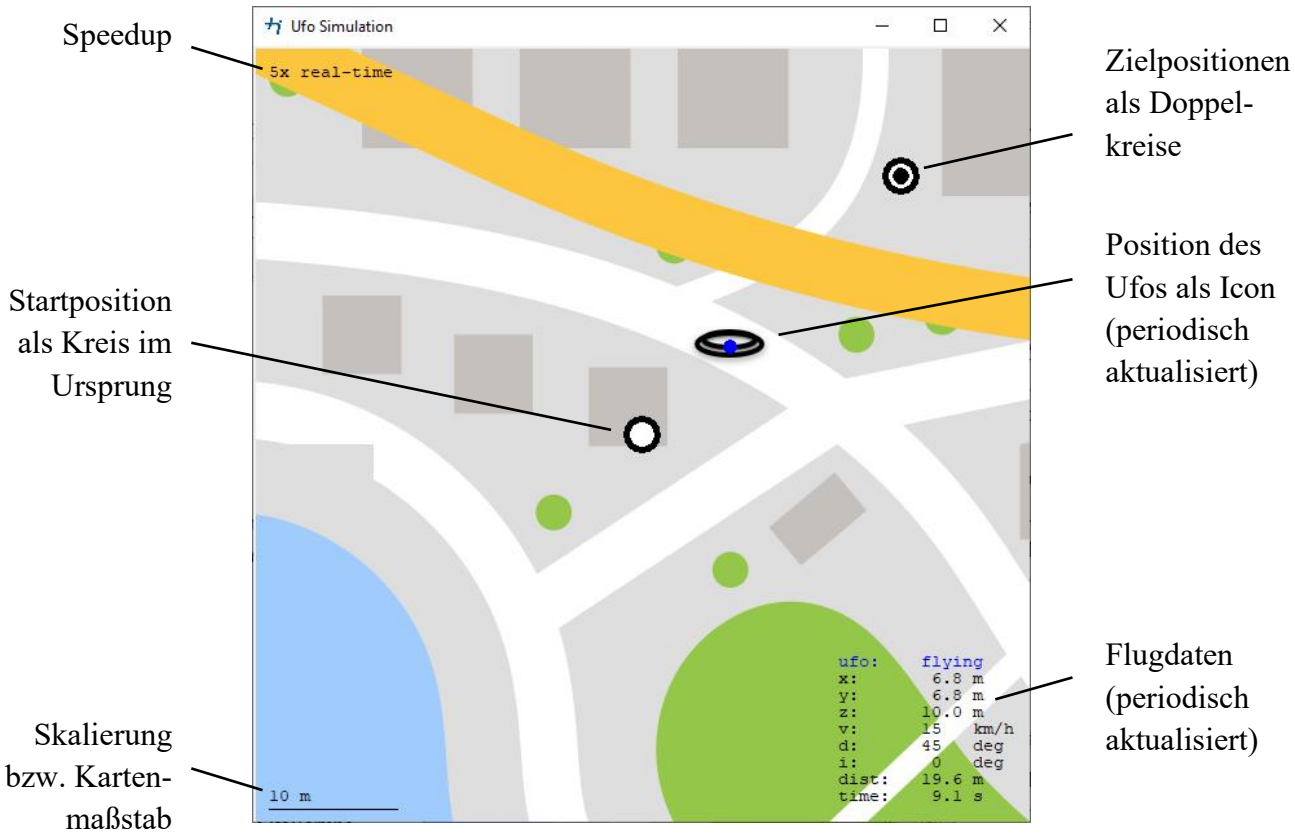

Abb. 6: View während des Fluges

Abb. 7: Anzeige der Neigung

Die Karte ist lediglich ein Hintergrundbild. Der Maßstab, d.h. Pixel pro Meter ändert sich bei Verkleinerung oder Vergrößerung des Fensters nicht. Es ist lediglich ein größerer Karten-ausschnitt sichtbar. Die Flugdaten und die Position des Ufos werden laufend aktualisiert. Während des Steigens und des Sinkens wird die Neigung in Grad angezeigt (Abb. 7).

Der Maßstab (Skalierung) wird durch den zweiten Parameter der Funktion start eingestellt. Erlaubt sind Werte zwischen 1 und 100. Werte außerhalb dieses Bereichs haben zur Folge, dass keine View geöffnet wird. Möchte man also ohne View fliegen, kann man als Skalierung z.B. den Wert 0 einsetzen.

Auch hier ist zu beachten ist, dass die Funktion start beim Aufruf zwei ganzzahlige Argumente und eine Liste verlangt. Ist beim Aufruf das erste oder das zweite Argument eine Gleitpunkt-zahl, kann sich ein Laufzeitfehler ergeben.