AERSP 424: Advanced Computer Programming

Homework 3 Spring 2023

Submission Instructions:

- Submit a .zip files containing your .cpp and/or .h files.
- Your code needs to be compilable.
- If you upload an updated submission, please remove the previous submission as only the final submission will be graded.
- Using comments to explain your code is mandatory.
- Submission deadline is at 11:59 PM on Friday 4/14/2023 (The late policy mentioned in the syllabus will be applied).
- Explicitly declare variables with a proper name (easy to understand) and datatype (reflect the real-world scenario if possible).

Question 1 (40 points): *Polymorphism and Virtual Functions*

So far, we have seen equations of motion for a 6-DOF rigid body. Though these equations are suitable for aerial vehicles, there are significant differences between each type of vehicles, for example, a multirotor, and an airplane. The major differences lie in the way forces and moments are generated between each vehicle platform. Consider the **'FlightSim'** class from the lecture, sample code, and HW2 solution. Add a *pure* virtual function named *set_actuators*() into this class. Then, derive three classes from this base class as follows.

A **'Multirotor'** class that will override the virtual function using the following equation.

$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ k_t & k_t & k_t & k_t \\ -lk_t & lk_t & lk_t & -lk_t \\ lk_t & lk_t & -lk_t & -lk_t \\ k_\tau & -k_\tau & k_\tau & -k_\tau \end{bmatrix} \begin{bmatrix} \Omega_1^2 \\ \Omega_2^2 \\ \Omega_3^2 \\ \Omega_4^2 \end{bmatrix}$$

where Ω_i is a rotational speed of each motor, k_t is a thrust coefficient, k_τ is a torque coefficient, and l is an arm length.

A **'Airplane'** class that will override the virtual function using the following equation.

$$\begin{bmatrix} F_x \\ F_y \\ F_z \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} X_{\delta_t} & X_{\delta_a} & X_{\delta_e} & X_{\delta_r} \\ Y_{\delta_t} & Y_{\delta_a} & Y_{\delta_e} & Y_{\delta_r} \\ Z_{\delta_t} & Z_{\delta_a} & Z_{\delta_e} & Z_{\delta_r} \\ L_{\delta_t} & L_{\delta_a} & L_{\delta_e} & L_{\delta_r} \\ M_{\delta_t} & M_{\delta_a} & M_{\delta_e} & M_{\delta_r} \\ N_{\delta_t} & N_{\delta_a} & N_{\delta_e} & N_{\delta_r} \end{bmatrix} \begin{bmatrix} \delta_t \\ \delta_a \\ \delta_e \\ \delta_r \end{bmatrix}$$

where δ_i is a change in thrust, aileron, elevator, and rudder, respectively. Each element of the matrix represents a stability derivative corresponding to each δ_i.

A **'ATC'** class that does not need to do anything with the virtual function. This class however contains a *static* function named *ADSB*(*some input arguments*) that can receive an object with either of these vehicle types and print out its information, i.e., whether the object is a multirotor or an airplane as well as the (x, y, z) position.

Perform iterations/simulation in the similar manner to HW1 and HW2 for an object instantiated from the **'Multirotor'** and the **'Airplane'** class with the controls (forces and moments) calculated from the above equations. In each iteration, each vehicle provides its information to the *ADSB(some input arguments)* function to print out.

Use the same initial states and parameters, and information for the table below. Pick any real number between 0 to 100 for Ω_i and between 0 to 1 for δ_i. Each Ω_i and δ_i must have a different value.

Variables	Value	Variables	Value	Variables	Value	Variables	Value
X_{δ_t}	1.0	X_{δ_a}	0.0	X_{δ_a}	0.0	X_{δ_r}	0.0
Y_{δ_t}	0.0	Y_{δ_a}	0.0	Y_{δ_a}	0.0	Y_{δ_r}	0.5

Z_{δ_t}	-0.5	Z_{δ_a}	0.0	Z_{δ_a}	-0.1	Z_{δ_r}	0.0
L_{δ_t}	0.0	L_{δ_a}	1.0	L_{δ_a}	0.0	L_{δ_r}	0.1
M_{δ_t}	0.01	M_{δ_a}	0.0	M_{δ_a}	1.0	M_{δ_r}	0.0
N_{δ_t}	0.0	N_{δ_a}	0.01	N_{δ_a}	0.0	N_{δ_r}	1.0
k_t	0.001	k_τ	0.0001	l	0.1		

Note:

- You may modify the base class encapsulation as you see fit.
- ATC = Air Traffic Control
- ADSB = Automatic Dependent Surveillance - Broadcast (ADS-B)

The following figure shows an example output.

```
Honework 3 Question 1
This is multirotor. My location (x,y,z) is (0, 0, -1)
This is airplane. My location (x,y,z) is (0, 0, -1)
This is multirotor. My location (x,y,z) is (0, 0, -0.999971)
This is airplane. My location (x,y,z) is (3.33333e-07, 2.66667e-08, -0.99999)
This is multirotor. My location (x,y,z) is (2.35261e-11, -1.99988e-10, -0.999912)
This is airplane. My location (x,y,z) is (1.00001e-06, 7.99931e-08, -0.999971)
This is multirotor. My location (x,y,z) is (1.09776e-10, -9.33234e-10, -0.999824)
This is airplane. My location (x,y,z) is (2.00004e-06, 1.59973e-07, -0.999942)
This is multirotor. My location (x,y,z) is (3.13607e-10, -2.66629e-09, -0.999706)
This is airplane. My location (x,y,z) is (3.33344e-06, 2.666e-07, -0.999904)
This is multirotor. My location (x,y,z) is (7.05521e-10, -5.99899e-09, -0.999559)
This is airplane. My location (x,y,z) is (5.00021e-06, 3.99869e-07, -0.999855)
This is multirotor. My location (x,y,z) is (1.37164e-09, -1.16645e-08, -0.999382)
This is airplane. My location (x,y,z) is (7.00036e-06, 5.59775e-07, -0.999798)
This is multirotor. My location (x,y,z) is (2.4137e-09, -2.05291e-08, -0.999177)
This is airplane. My location (x,y,z) is (9.33391e-06, 7.46314e-07, -0.99973)
This is multirotor. My location (x,y,z) is (3.94898e-09, -3.35926e-08, -0.998941)
This is airplane. My location (x,y,z) is (1.20009e-05, 9.59481e-07, -0.999653)
This is multirotor. My location (x,y,z) is (6.11029e-09, -5.1988e-08, -0.998677)
This is airplane. My location (x,y,z) is (1.50012e-05, 1.19927e-06, -0.999566)
This is multirotor. My location (x,y,z) is (9.04596e-09, -7.69817e-08, -0.998382)
This is airplane. My location (x,y,z) is (1.8335e-05, 1.46569e-06, -0.99947)
This is multirotor. My location (x,y,z) is (1.29197e-08, -1.09973e-07, -0.998059)
This is airplane. My location (x,y,z) is (2.20022e-05, 1.75873e-06, -0.999364)
This is multirotor. My location (x,y,z) is (1.79108e-08, -1.52495e-07, -0.997706)
This is airplane. My location (x,y,z) is (2.60029e-05, 2.07838e-06, -0.999248)
This is multirotor. My location (x,y,z) is (2.42135e-08, -2.06214e-07, -0.997324)
This is airplane. My location (x,y,z) is (3.0337e-05, 2.42465e-06, -0.999123)
```

Question 2 (30 points): *Libraries*

- Pick one or more *non-standard opensource* C++ libraries you like (either from examples during lectures or somewhere else).

- Run at least two examples using those libraries that you use (either two examples from the same library or one example from each library, or more). You can pick examples from the website or write your own code using those libraries.
- You can also write your own library as *one of the two* libraries. The other library must be a 3^{rd}-party opensource library.
- Look into the library source code of the function(s) that you call in the examples.
- Add `std::cout << "Your first name and your last name" << std::endl;` at the beginning of the function(s). Replace the string with your actual first name and last name.
- Rebuild / recompile the library.
- Run the examples again.
- Provide a link to the website of the 3^{rd}-party opensource libraries you are using in the submission as well.

Question 3 (30 points): *Multithreading / Multicore programming*

From Question 1, the multirotor and the airplane are independent, so the calculation of their state information should be done concurrently.

Write a multithreading/multicore program with two threads that performs the calculation, e.g., set_init_state, set_param, set_actuators, dynamics, integration, constrainAngles, and ADSB, for each vehicle simultaneously.

Create an additional thread that will receive information from those two threads and compute the distance of each vehicle from the origin. Additionally, the first two threads should have lines of code or a function that send their information to the third thread.

All these should be done in a thread-safe manner.

Note:

- You may or may not use a ThreadSafeQueue header file uploaded in the sample code folder on Canvas.

- You may need to delay the beginning of the third thread for a little bit in order for it to receive an information, e.g.,

```
#include <thread>
#include <chrono>
std::this_thread::sleep_for( std::chrono::seconds( 1 ) );
```

The following figure shows an example of the non-thread-safe behavior.

```
This is This is multirotor. My location (x,y,z) is (-40.1784, -12.0349, airplane. My location (x,y,z) is (0.4996, 0.395994, 115.07).41.3738).
This is
airplaneThis is multirotor. My location (x,y,z) is (-40.2169, -12.043, . My location (x,y,z) is (0.499783, 0.396096, 41.4026).
This is 115.144).airplane. My location (x,y,z) is (0.499965, 0.396199,
41.4314).
-40.2169, -12.043, This is airplaneThis is . I am airplane115.144, 1 meters away from the origin.. My location (x,y,z) is (0.500148,
8.53454, 20.4888, 81.3949, 0.396302, This is multirotor. I am 10.725514, -0.0205581, -1.5405, 41.4601). meters away from the origin.

-2.91417, 3.01201, This is airplaneThis is . I am 0.99999 meters away from the origin.airplane. My location (x,y,z) is (-29.1526,
This is 0.500331,
multirotor. I am 0.999971 meters away from the origin.0.396405,
This is airplane41.4889).. I am
0.999971 meters away from the origin.This is
This is airplane. My location (x,y,z) is (multirotor0.500513, . I am 0.396509, 0.999912 meters away from the origin.41.5177).
This is
airplane. I am This is 0.999942 meters away from the origin.airplane. My location (x,y,z) is (0.500696, 0.396613,
This is multirotor41.5465).. I am 0.999824 meters away from the origin.

This is This is airplaneairplane. My location (x,y,z) is (0.500879, . I am 0.999904 meters away from the origin.0.396717, 41.5754).
This is
multirotor. I am This is airplane0.999706 meters away from the origin.. My location (x,y,z) is (0.501062,
This is airplane. I am 0.999855 meters away from the origin.0.396822,
41.6042).
This is multirotor. I am 0.999559 meters away from the origin.
This is airplane. My location (x,y,z) is (This is airplane0.501245, 0.396927, . I am 0.999798 meters away from the origin.
41.633).This is multirotor
. I am 0.999382 meters away from the origin.
This is airplaneThis is airplane. I am 0.99973 meters away from the origin.
. My location (x,y,z) is (0.501428, This is multirotor0.397032, . I am 0.999177 meters away from the origin.41.6619).
This is airplane. I am 0.999653 meters away from the origin.
```

The following figure shows an example of the thread-safe behavior.

```
This is multirotor. My location (x,y,z) is (-3.03191, -2.308, 16.5823).
This is multirotor. My location (x,y,z) is (-3.03933, -2.31171, 16.612).
This is multirotor. My location (x,y,z) is (-3.04676, -2.31542, 16.6419).
This is multirotor. My location (x,y,z) is (-3.05421, -2.31914, 16.6717).
This is multirotor. My location (x,y,z) is (-3.06167, -2.32286, 16.7016).
This is multirotor. My location (x,y,z) is (-3.06915, -2.32659, 16.7314).
This is multirotor. My location (x,y,z) is (-3.07665, -2.33032, 16.7613).
This is multirotor. My location (x,y,z) is (-3.08416, -2.33405, 16.7913).
This is multirotor. My location (x,y,z) is (-3.09169, -2.33779, 16.8212).
This is multirotor. My location (x,y,z) is (-3.09923, -2.34154, 16.8512).
This is multirotor. My location (x,y,z) is (-3.10679, -2.34528, 16.8812).
This is multirotor. I am 5.55951 meters away from the origin.
This is multirotor. I am 5.57799 meters away from the origin.
This is multirotor. I am 5.59649 meters away from the origin.
This is multirotor. I am 5.61501 meters away from the origin.
This is multirotor. I am 5.63355 meters away from the origin.
This is multirotor. I am 5.65211 meters away from the origin.
This is multirotor. I am 5.6707 meters away from the origin.
This is multirotor. I am 5.68931 meters away from the origin.
This is airplane. I am 0.861547 meters away from the origin.
This is airplane. I am 0.859908 meters away from the origin.
This is airplane. I am 0.858261 meters away from the origin.
```