ZERO ROBOTICS

SPHERES ISS CHALLENGE

Spinning: Angular Velocity

ZERO ROBOTICS
SPHERES ISS CHALLENGE

Rotational Kinematics

- At this point, you are familiar with linear kinematics, including linear (tangential) velocity.
- We will now discuss **rotational kinematics**, specifically **angular velocity** (**ω**).
- In this tutorial you will:
  - learn the basics of angular velocity
  - use the function setAttRateTarget to spin by controlling angular velocity
  - learn the effects of angular momentum on rotating bodies

Angular Velocity

- Angular velocity is in terms of **angular displacement** (θ).
- The components of angular velocity (ωx, ωy, and ωz) are stored in the last three elements of your state array.
- Like linear velocity, angular velocity is a vector.
- The magnitude of angular velocity is intuitive: it is equal to angular speed, or how many radians are covered per unit time.
  - ex: π/6 radians per second
- The direction of the angular velocity vector follows the same principle as the direction of torque.

Just Like Torque

- Although you are spinning, the direction of angular velocity is constant.
- The direction of an angular velocity vector is always **normal**, or perpendicular, to the plane of rotation. So, the angular velocity vector is always parallel to the axis of rotation.
- For example, if you set a positive value for ωz, the SPHERE will rotate counter-clockwise on its z-axis. If this sounds unfamiliar, review the tutorial on setTorques.

setAttRateTarget

- **setAttRateTarget** is another function you can use to make your SPHERE spin. setAttRateTarget is to setTorques as setVelocityTarget is to setForces. It's closed loop, aims for a target value, and is more stable.
- Why, then, did we teach you setTorques? setTorques is still a valid and useful function, and is a basic way to show that torque causes rotational motion.
- setAttRateTarget controls angular velocity, so you can control how fast your SPHERE spins. Also, because it is closed loop, there is relatively little wobble.

Let's Try It

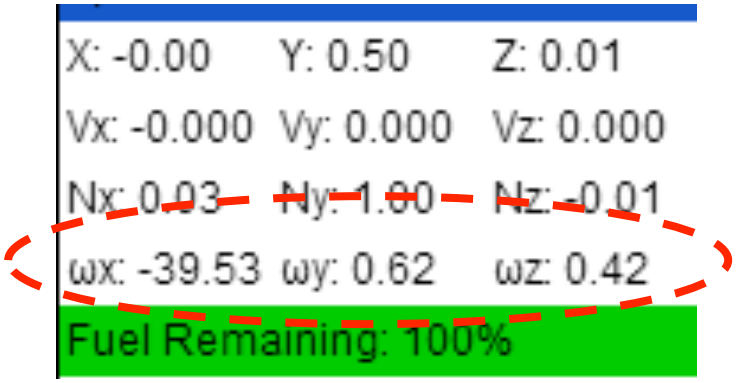
- In this scenario, we want to spin:
  - clockwise on the body yz-plane
  - at a rate of 40° per second.
- Create a new project (Project17) and create an array called **targetRate**. We will show you how to initialize the components of targetRate on the next slide. Try to figure it out yourself, then check your solution.
- Keep in mind that all specifications must be in radians. To convert degrees to radians, multiply the value in degrees by π/180.

Initialize targetRate

- Spinning on the body yz-plane means spinning on the body x-axis. Since we want to spin clockwise, we need a negative value for ωx.
- Initialize targetRate and set the SPHERE's attitude rate target to targetRate. Compile and run.

```
1  float targetRate[3];
2
3  void init(){
4    targetRate[0]=-40*PI/180;
5    targetRate[1]=0.00;
6    targetRate[2]=0.00;
7  }
8
9  void loop(){
10   api.setAttRateTarget(targetRate);
11 }
```

Results



- In the display panel, angular velocity is written in degrees. The satellite hits the target rotation rate and maintains it for the duration of the session.
- Also, because setAttRateTarget is closed loop, the satellite is very stable while spinning.

Keep Going

- Let's take it a step further. Create an int **counter**, initialize it to 0, and increment it at the last line of loop.
- Write a conditional so that setAttRateTarget will only be called during the first 20 seconds of the game. Compile and run.

```
1  float targetRate[3];
2  int counter;
3
4  void init(){
5    targetRate[0]=-40*PI/180;
6    targetRate[1]=0.00;
7    targetRate[2]=0.00;
8
9    counter=0;
10 }
11
12 void loop(){
13   if (counter<20)
14     api.setAttRateTarget(targetRate);
15   counter++;
16 }
```

Results

- After 20 seconds, the SPHERE continues to spin due to Newton's First Law.
- As time passes, the spin becomes less stable due to changes in **angular momentum**.
- Angular momentum is equated by:

$$L = I\omega$$

- *I* represents **moment of inertia**. Inertia is essentially a measure of resistance to change in motion.
- Angular momentum increases with inertia and angular velocity. In competitions in which the mass of your satellite changes, know that inertia increases with mass.

Modification

- Remember that in microgravity, there are no forces like friction and air resistance to bring objects to rest. SPHERES stop by firing their thrusters in the opposite direction.
- After 20 seconds, no thrusters fired to stop the satellite from spinning. The satellite continued to spin because it had a great deal of angular momentum and faced no opposing forces.
- In order to stop spinning, we need to write a solution that decreases angular velocity after 20 seconds. There are several ways to do this; we will show you the simplest method.

Stop

- Create a new array **stop** and initialize it with zeroes for all three components.
- Write an else statement that sets the attitude rate target to stop after 20 seconds.
- Compile and run.

```
10    stop[0]=0.00;
11    stop[1]=0.00;
12    stop[2]=0.00;
13
14    counter=0;
15  }
16
17  void loop(){
18    if (counter<20)
19      api.setAttRateTarget(targetRate);
20    else
21      api.setAttRateTarget(stop);
22    counter++;
23  }
```

Results

- It works! It takes a few seconds, but the satellite stops spinning.
- If you continue to observe the satellite, you will notice that it makes small movements and rotates slightly in an arbitrary direction. This is because it is constantly firing its thrusters in attempt to stop all angular motion.
- You can prevent this by changing the “else” to an “else if” dependent on time.
- As always, it’s up to you to create the most efficient and innovative algorithm. Spinning is required for almost every ZR competition, so creating a versatile function will help you from year to year.