Spinning: Torque
Rotational Motion

• At this point, you are pretty well versed in translational motion. It’s now time to get into rotational motion.

• We’ll start our focus on rotational motion by learning how to spin. You already know how to rotate to face a direction, but you don’t know how to continuously change your attitude yet. To do this, we can use the function `setTorques`.

• In this tutorial you will:
  – learn the basics of rotational motion and torque
  – practice using body frame controls
  – control direction of spin with cross products and the right hand rule
• **Torque** is the rotational equivalent of force. It causes rotational motion and **angular acceleration**.

• Like setForces, setTorques is an open loop control function. setTorques doesn’t use a target value; it sets the rotational impulses to be applied to the satellite every time the thrusters fire.

• Also like setForces, setTorques can be used with closed loop control functions, but be very careful.
Axis of Rotation

• All of the functions we have covered so far operate on the **global frame**, meaning that the x-, y-, and z-components refer to the axes of the coordinate plane.

• setTorques operates on the **body frame**, meaning that the x-, y-, and z-components refer to the axes of the satellite.
SPHERES Faces

+X

-X

+Y

-Y

+Z

-Z
• Torque ($\tau$) is equal to the cross product of force and radius.

$$\tau = F \times r$$

• The cross product of two vectors is a resultant vector that is perpendicular to both original vectors. We will revisit cross product later.

• For now, just know that torque is always parallel to the axis of rotation.
Axis of Rotation

• You will need to know how to control the direction of spin. As we’ve mentioned before, setTorques operates on the body frame.

• Associate a **positive** value with **counter-clockwise** rotation.

• Associate a **negative** value with **clockwise** rotation.

• See the next slide for an example.
Axis of Rotation

• For example, if we set a positive value for the z-component of torque, the satellite will rotate counterclockwise on its z-axis like this:

• We’ll show you a few tips later to help you better understand direction of spin. But first, let’s code!
Try It

• Create a new project called Project16. Create an array called `torques` and initialize it with a very small value for the z component.

```c
float torques[3];
void init(){
    torques[0]=0.00;
    torques[1]=0.00;
    torques[2]=0.01;
}
```

• `setTorques` to `torques` and run the simulation.

```c
void loop(){
    api.setTorques(torques);
}
```
Results

• We expect the satellite to rotate counter-clockwise on its z-axis because we set a positive z-component value.
• One way to test this is to rotate the coordinate plane so that the +Z face of the SPHERE is facing you (see Slide 6). It might help to zoom in.
• The satellite is clearly rotating counter-clockwise, so our code passes.
Results

body z-axis
(axis of rotation)

+Z face

coordinate y-axis

counter-clockwise rotation
Right Hand Rule

• Another way to test if your satellite is spinning the right way is to use the **right hand rule**. To understand the right hand rule, we will need to revisit cross products.

• Torque is the cross product of force and radius. The right hand rule lets us visualize this with our fingers.
Right Hand Rule

• Your thumb represents the radius of the SPHERE.
• Your index finger represents the direction the SPHERE is turning.
• Your middle finger represents torque.
• Let’s try this for our simulation. Before you see the diagram on the next page, try it yourself.
• Rotate your hand so it lines up with the satellite. The rotational force spins the satellite counter-clockwise, so torque should point out of the +Z face.
Right Hand Rule
Stabilization

• As you may have noticed, rotation using setTorques is unstable. The wobble is distinctly noticeable around 15 seconds, and the spin becomes increasingly shaky as time progresses.

• Before you start thinking about ways to stabilize your spin algorithm, go on to the next tutorial on angular velocity. Good job for making it this far!