ZERO ROBOTICS

SPHERES ISS CHALLENGE

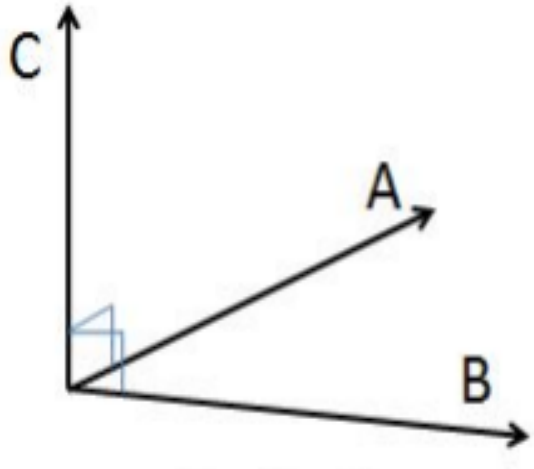
Spinning: Tilted Axis

ZERO ROBOTICS
SPHERES ISS CHALLENGE

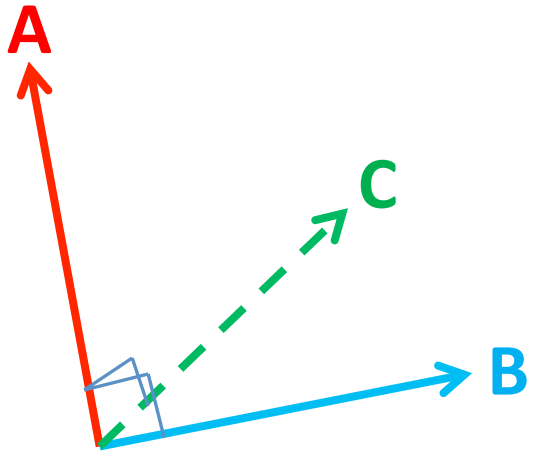
Tilted Axis

- Up to this point, we've been spinning by setting rotation rates for just one component. But, the axis of rotation won't always be straight up and down. Tilted axes make it difficult to set the target rotation rates because they usually involve all three components.
- One way to handle tilted axes is to use cross product. Remember that the cross product of two vectors is a resultant vector perpendicular to both original vectors.
- This tutorial will teach you to use cross product to make your SPHERE spin on any axis at an arbitrary angular velocity.

Cross Product



A x B = C

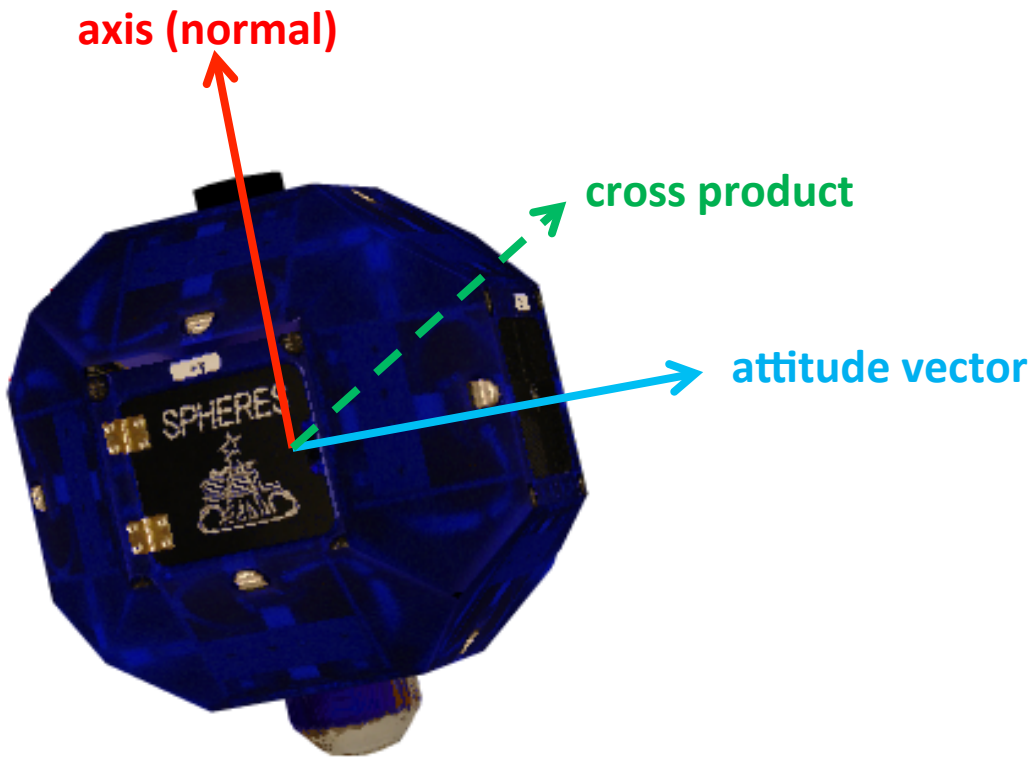


A x **B** = **C**

Cross Product

- Crossing the SPHERE's attitude vector with the axis of rotation (also known as the normal to the plane of rotation) will yield a resultant attitude vector pointing 90° degrees away on the plane of rotation.
- Every time the satellite moves, the resultant vector moves – there will always be 90° between the SPHERE's attitude vector and the resultant attitude vector. So, if we set our attitude target to the cross product in loop(), we will spin.

SPHERES Diagram



Calculating Cross Product

- To take the cross product of two vectors [a, b, c] and [d, e, f] by hand, you will need to use matrices.

[a, b, c] x [d, e, f] =

$$\begin{vmatrix} x & y & z \\ a & b & c \\ d & e & f \end{vmatrix} = \begin{vmatrix} b & c \\ e & f \end{vmatrix} x - \begin{vmatrix} a & c \\ d & f \end{vmatrix} y + \begin{vmatrix} a & b \\ d & e \end{vmatrix} z =$$

[(bf-ce), -(af-cd), (ae-bd)]

Matrices

- If you are familiar with matrices, you will see that we took the determinant of a 3x3 matrix. The first line is the coordinate axes, and the second and third lines are the vectors we are crossing.
- The easiest way to do this is to break the matrix into cofactor matrices and take determinant of each of those 2x2 matrices.
- If you don't have experience with matrices, you may find it easiest to substitute the components of your vectors into the formula in bold on the previous slide.

The Scenario

- Luckily, the API has a function called **mathVecCross** that calculates the cross product for you. Let's test it out.
- In this example, our goal is to rotate on a random axis. We aren't concerned with whether its clockwise or counterclockwise, nor will we attempt to control angular velocity.
- Create a new project called Project19 and create four arrays: **myState**, **myAtt**, **targetAtt**, and **axis**.
- Initialize the components of axis with random numbers. We chose [0.4, 0.3, 0.7].

Prep Work

- Before we call mathVecCross, we need to do a few things in loop().
- First and foremost, your axis vector needs to be normalized.
- Retrieve myState and fill myAtt with the components of your attitude vector.

```
12 void loop(){
13   mathVecNormalize(axis,3);
14   api.getMyZRState(myState);
15   for (int i=0; i<3; i++)
16     myAtt[i]=myState[i+6];
```

mathVecCross

- Now we can find the cross product of myAtt and axis. Use mathVecCross to calculate the cross product and store it in targetAtt.
- Normalize targetAtt and make it your attitude target.
- Compile and run.

```
18    mathVecCross(targetAtt,myAtt,axis);
19    mathVecNormalize(targetAtt,3);
20    api.setAttitudeTarget(targetAtt);
21  }
```

Results

- The SPHERE spins, but it is very unstable.
- The way we have coded it, the attitude target is set to the cross product of our current attitude and the axis. One problem with this is that any error in our current attitude is amplified. If our attitude is even a fraction of a degree off the plane of rotation, the target attitude will also be off, and the snowball effect occurs.
- Another issue is that we don't give the SPHERE enough time to rotate. We can't expect it to spin at 90 deg/sec, so it will never reach the target attitude.

Modifications

- The tutorial ends here. There are many ways to stabilize spinning; try to find the best.
- Along with stabilization, you should also focus on angular velocity. Think of ways to combine mathVecCross with setAttRateTarget and other functions you know.
- Be creative, think outside the box, and test your algorithms! An inefficient algorithm that works 100% of the time is better than an efficient algorithm that only works 50% of the time.