Force
setForces

- When you set a position, velocity, or attitude target, you are controlling forces in a **closed loop** system. This means that the satellite auto-adjusts its forces to meet your target.

- In addition to the closed functions we have covered so far, you can directly control force with **setForces**. This is an **open loop** control function, meaning the satellite will NOT self-adjust its forces. You need to continuously provide new input. On the bright side, this is very easy with `loop()`, and we are used to providing continuous input.

- `setForces` delivers the specified amount of force as impulses to the satellite every time the thrusters are fired. Unlike `setVelocityTarget` and other functions that aim for a target value, `setForces` has no target. You need to control the amount of force delivered with your code.
The Scenario

• In this example, we will continue our quest to move to a position target as quickly as possible.
• Open Project14c from the last tutorial and save it as Project15. It should look like this:

```c
float item[3];
float myState[12];
float myPos[3];
float vectorBetween[3];
float distance;

void init(){
    item[0]=0.8;
    item[1]=0.0;
    item[2]=0.0;
}

void loop(){
    api.getMyZRState(myState);
    for (int i=0; i<3; i++)
        myPos[i]=myState[i];
    mathVecSubtract(vectorBetween, item, myPos, 3);
    distance = mathVecMagnitude(vectorBetween, 3);
    if (distance>0.6)
        api.setVelocityTarget(vectorBetween);
    else
        api.setPositionTarget(item);
}
```
Let’s recap what the code from the setVelocityTarget tutorial does.

Every second, we find the vector that points from our satellite to the item and store it in vectorBetween.

vectorBetween is really a distance vector, so we find the magnitude and store it in distance.

If distance is greater than 0.6 m, we set the velocity target to vectorBetween.

Otherwise, we set the position target to item so we have enough time to slow down.
Variable Target

• vectorBetween varies directly with distance. As the satellite approaches the target, the components of vectorBetween approach 0.

• So, our target velocity decreases with time. We want to use this principle for force as well. Simply change setVelocityTarget to setForces.

• Compile and run.

```plaintext
if (distance>0.6)
    api.setForces(vectorBetween);
else
    api.setPositionTarget(item);
```
Results

• It takes roughly 26 seconds for the satellite to stop on the item. Even then, the satellite adjusts itself at a very low speed for the next few seconds.

• This is better than our setPositionTarget time of 28 seconds, but doesn’t beat our setVelocityTarget time.

• You can improve this time quite a bit by adjusting the magnitude and conditional distance, but you may still find it easier to use setVelocityTarget.

• You can use setForces in tandem with setPositionTarget and setVelocityTarget, but be careful. Combining open loop and closed loop control functions can produce unanticipated results.