ZERO ROBOTICS

ISS PROGRAMING CHALLENGE

# Hints about SPHERES Loop Dynamics

# Goals

- In this tutorial you will look at:
  - –SPHERES dynamics related to Newton’s First Law
  - –Test out 4 different “What if?” Scenarios to see how your code can impact SPHERES dynamics

What if?

What if?

What if?

What if?

- Keep this tutorial in mind
  - –As you begin to program for the game
  - –As you review your game simulations
  - –As you troubleshoot your program

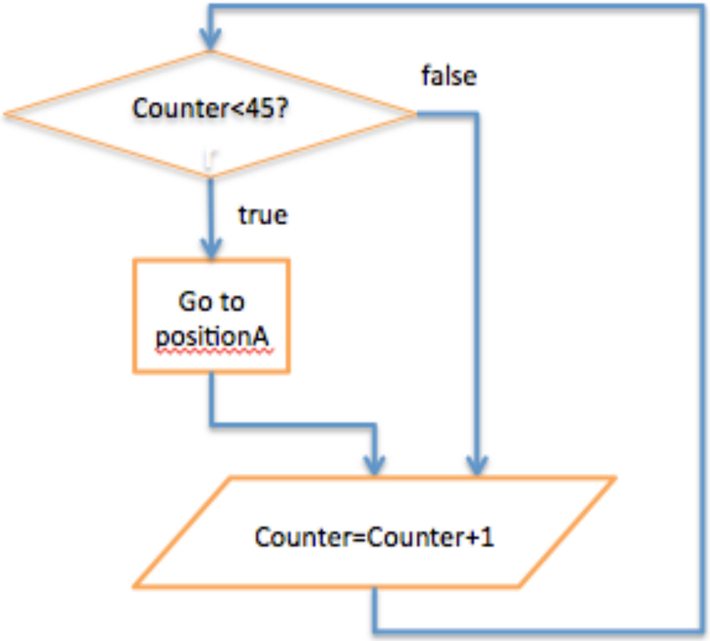
# Newton’s First Law and SPHERES

- ***First Law (The Law of Inertia):*** An object at rest remains at rest until acted on by an outside force; an object in motion remains in motion until acted on by an outside force.

- ***SPHERES Dynamics:*** The SPHERES thrusters release compressed $CO_2$ to create the forces that are used both to:
  - –Start the SPHERES motion
  - –Stop the SPHERES motion

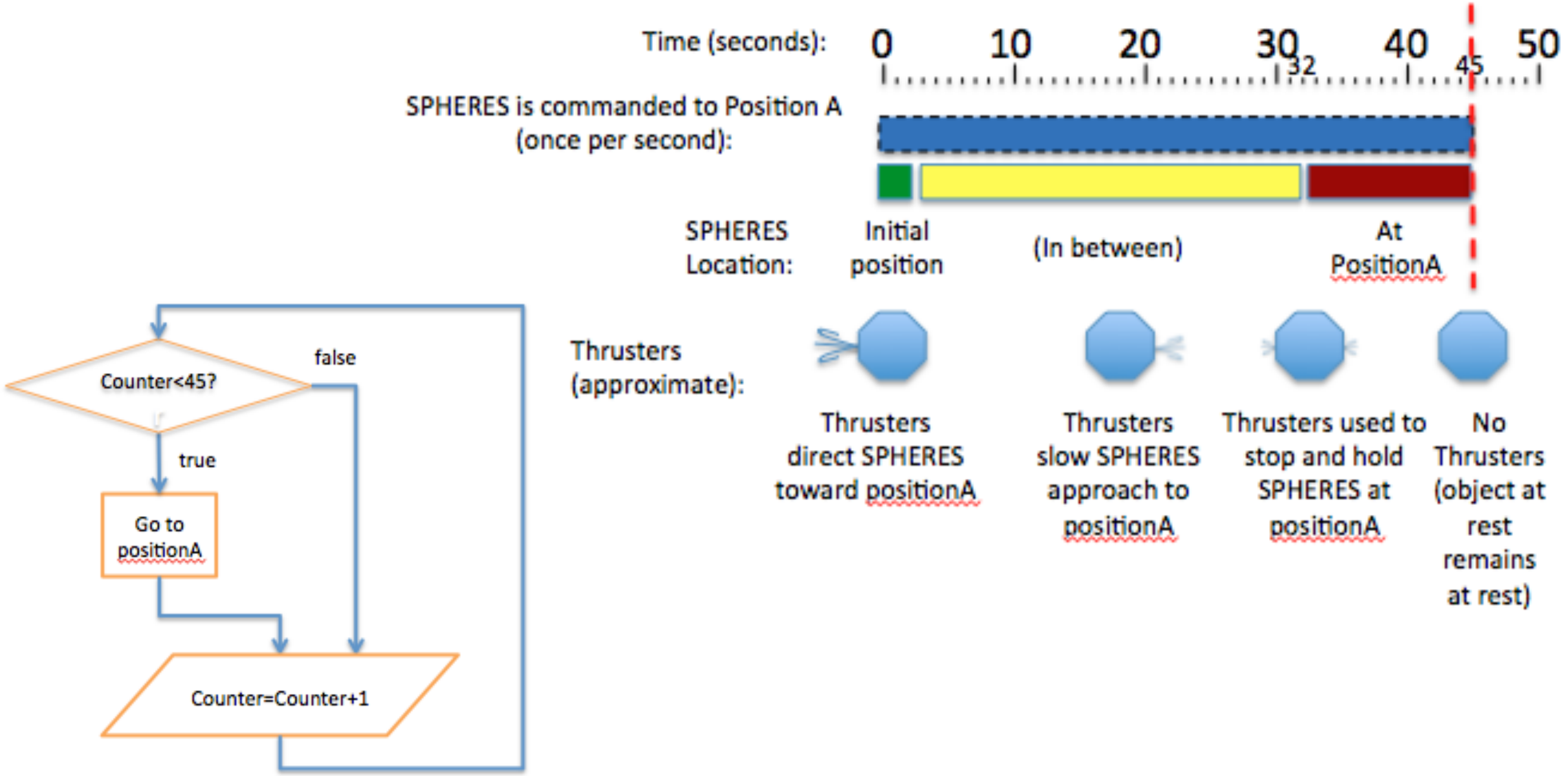
# Newton’s First Law and SPHERES, continued

- Let’s review how the SPHERES motion is controlled
- When your program repeatedly commands the SPHERES to move to a point (as shown in the loop on the right):
  - The satellite activates its thrusters to create a force that will move it in the direction of the point.
  - As the satellite nears the point it will activate other thrusters to start to slow itself down
  - Once the satellite reaches the point, it will activate thrusters to stop itself in place
  - When no longer commanded, the satellite will stop activating its thrusters



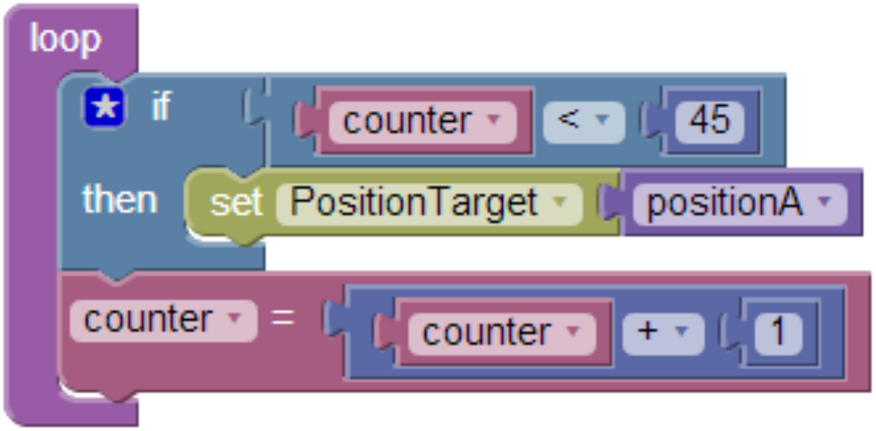
# Newton’s First Law and SPHERES, continued

- This process is described in the picture below

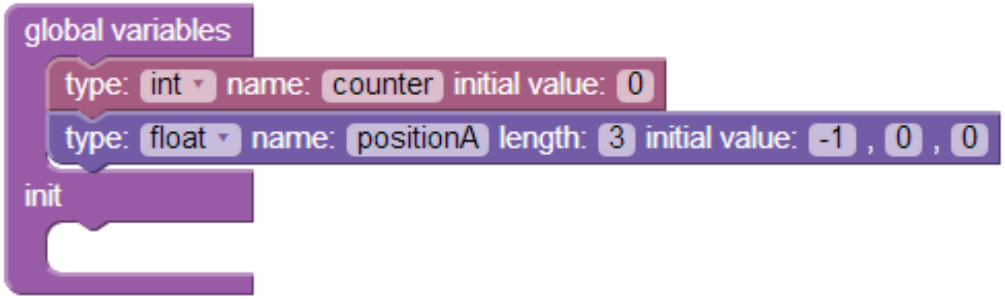

Time (seconds):
0
10
20
30
32
40
45
50
SPHERES is commanded to Position A (once per second):
SPHERES Location:
Initial position
(In between)
At PositionA
Thrusters (approximate):
Thrusters direct SPHERES toward positionA
Thrusters slow SPHERES approach to positionA
Thrusters used to stop and hold SPHERES at positionA
No Thrusters (object at rest remains at rest)
Counter<45?
false
true
Go to positionA
Counter=Counter+1

# Create a New Program

- We will create the simple program shown to the right to:
  - Demonstrate SPHERES dynamics
  - Test out 4 different “what-if?” scenarios
- First you need to create a new project:
  - Name it “dynamics” and choose “FreeMode” and “Graphical Editor”
  - Create the following variables and arrays:
    - int counter
    - float positionA[3]
      - Set initial value to (**-1,0,0**)



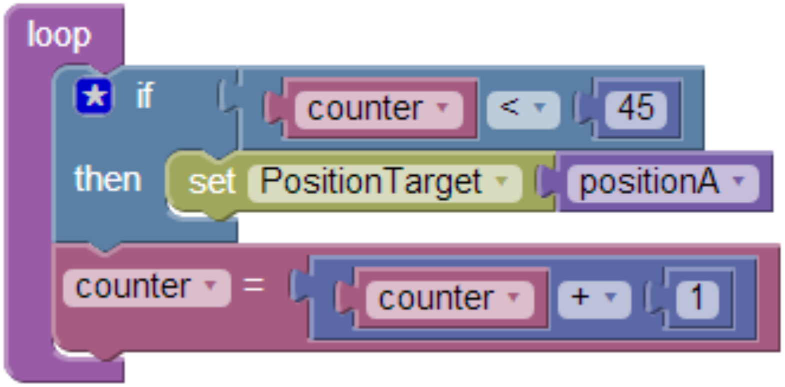
```
loop
    if   counter < 45
    then set PositionTarget positionA
    counter = counter + 1
```



```
global variables
    type: int   name: counter   initial value: 0
    type: float name: positionA length: 3 initial value: -1 , 0 , 0
init
```

# Create a New Program, continued

- Complete the program as shown
  - Hints for “If-then” statement
    - Drag an “If-then” statement into the loop from the logic accordion
    - Drag ”__ **==**__″ from the logic accordion and set it to “<“
    - Drag **counter** (“—Select--” block) from variable accordion and a number from the math accordion (45)
    - Drag a **setPostionTarget** block from the SPHERES Controls accordion into the If—then” block (set to positionA)
  - Hints for counter
    - Drag pink “Select=0” block from the variables accordion and toggle to “counter=0”
    - Drag ”__ **+**__″ from the math accordion
    - Drag **counter** from the variables accordion and a number (1) from math

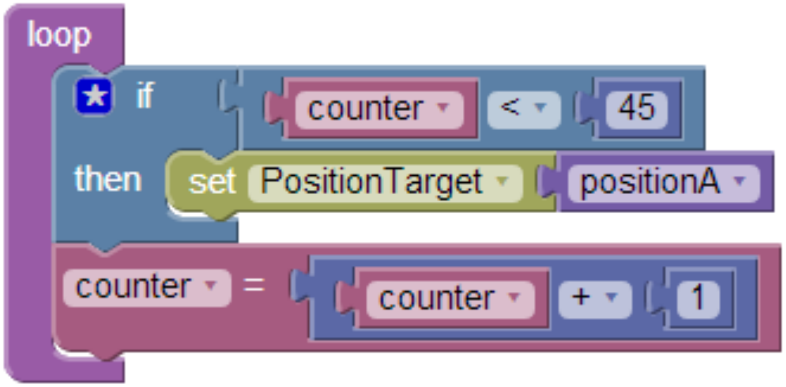


```
loop
  if counter < 45
  then set PositionTarget positionA
  counter = counter + 1
```

# Expected Dynamics

- Test your program!
  - Compile, Simulate
    - Maximum Time: 90s
  - View simulation at 2x speed

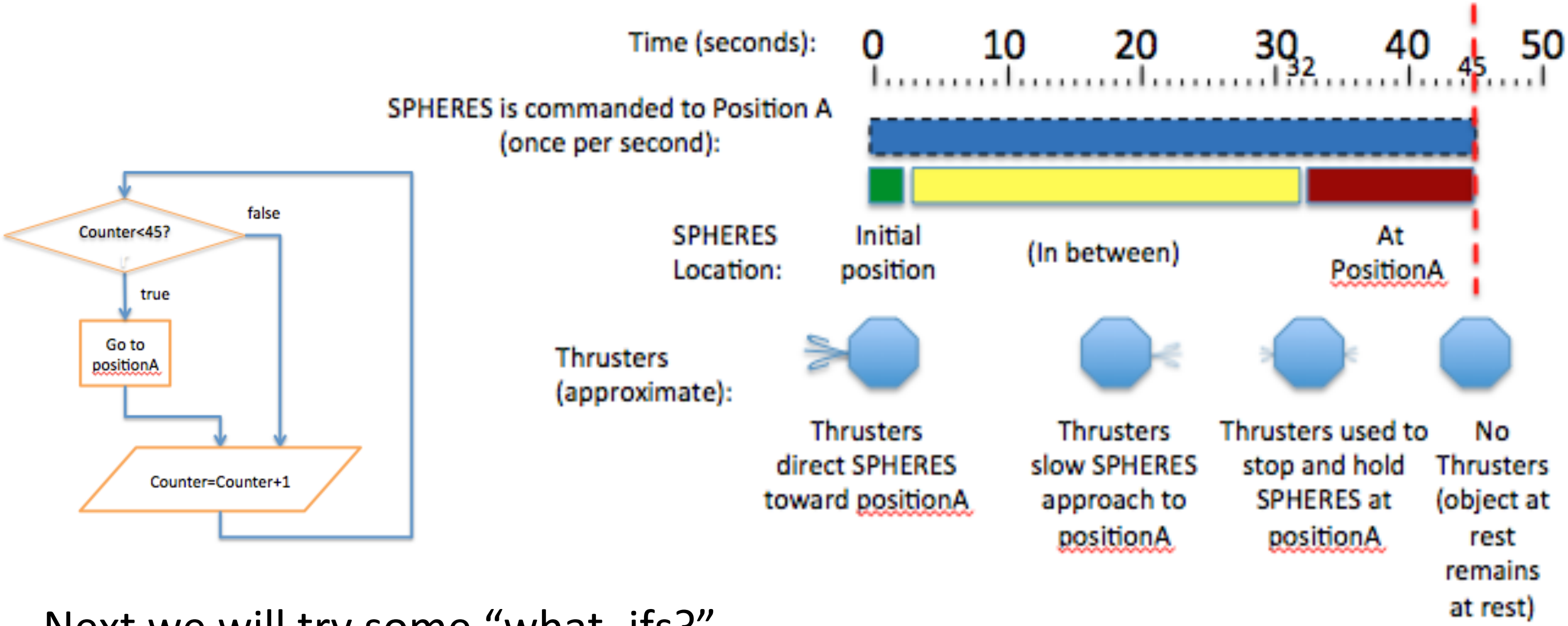
- The SPHERE should move to the point (-1,0,0) and stop there.
- Close simulation window.



```
loop
  if counter < 45
  then set PositionTarget positionA
  counter = counter + 1
```

# Expected Dynamics, continued

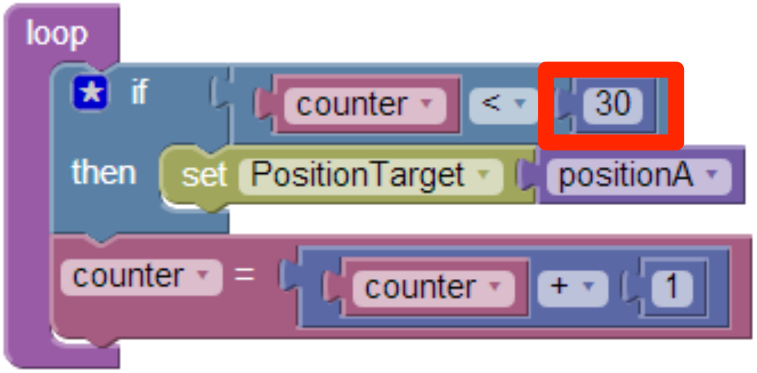
- Take another look at the SPHERES Dynamics depicted in the figure below
- Remember that the SPHERES reads the code in the loop once per second. For this example, this means the counter increases once per second
- The SPHERES reaches positionA near time = 32 seconds and stays at positionA, even after the counter reaches 45


Counter<45?
false
true
Go to positionA
Counter=Counter+1
Time (seconds):
0
10
20
30
32
40
45
50
SPHERES is commanded to Position A (once per second):
SPHERES Location:
Initial position
(In between)
At PositionA
Thrusters (approximate):
Thrusters direct SPHERES toward positionA
Thrusters slow SPHERES approach to positionA
Thrusters used to stop and hold SPHERES at positionA
No Thrusters (object at rest remains at rest)

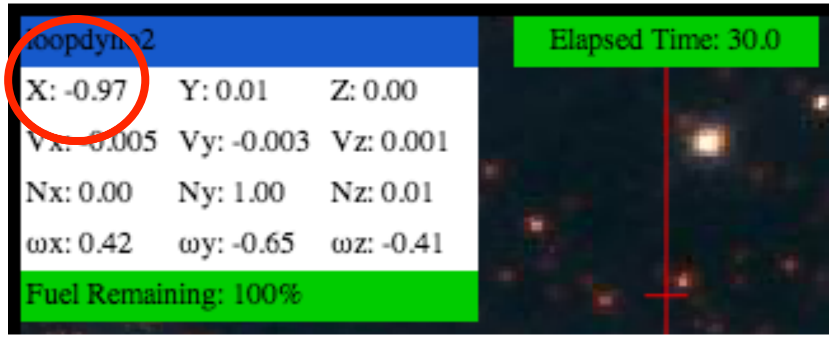
- Next we will try some “what–ifs?”.....

# What–if? #1

- What if we set counter<30 (instead of <45)?
- Test your program!
  - Compile, Simulate
    - Maximum Time: 90 seconds
  - View simulation at 2x speed
- Notice that the SPHERES slows down as it nears the point (-1,0,0) but keeps moving very slowly?
- What happened?
  - Just before the SPHERES reached “positionA” (-1,0,0) the conditional statement (counter<30) was no longer true (see image)



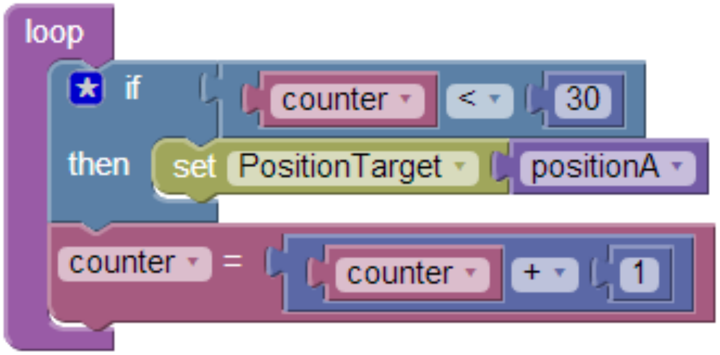
```
loop
  if    counter < 30
  then  set PositionTarget positionA
  counter = counter + 1
```



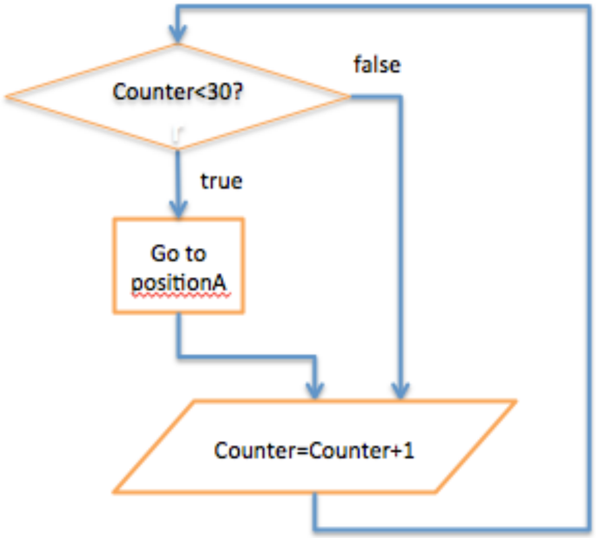
```
loopdyno2                               Elapsed Time: 30.0
X: -0.97      Y: 0.01      Z: 0.00
Vx: -0.005    Vy: -0.003   Vz: 0.001
Nx: 0.00      Ny: 1.00     Nz: 0.01
ωx: 0.42      ωy: -0.65    ωz: -0.41
Fuel Remaining: 100%
```

# What–if? #1 explained

- So why did the SPHERES continue to move?
- You can explain what happened using Newton's laws
  - Notice that when "counter<30?" is false the program does not contain any more SPHERES Control commands (see flow diagram)
  - Without commands, the thrusters shut off.
  - In this example the thrusters were shut off just before the SPHERES was fully stopped
  - "An object in motion remains in motion unless acted on by a force"
  - Since there is essentially no friction the SPHERES will continue to move at the same velocity it was moving when the thrusters were shut off!!

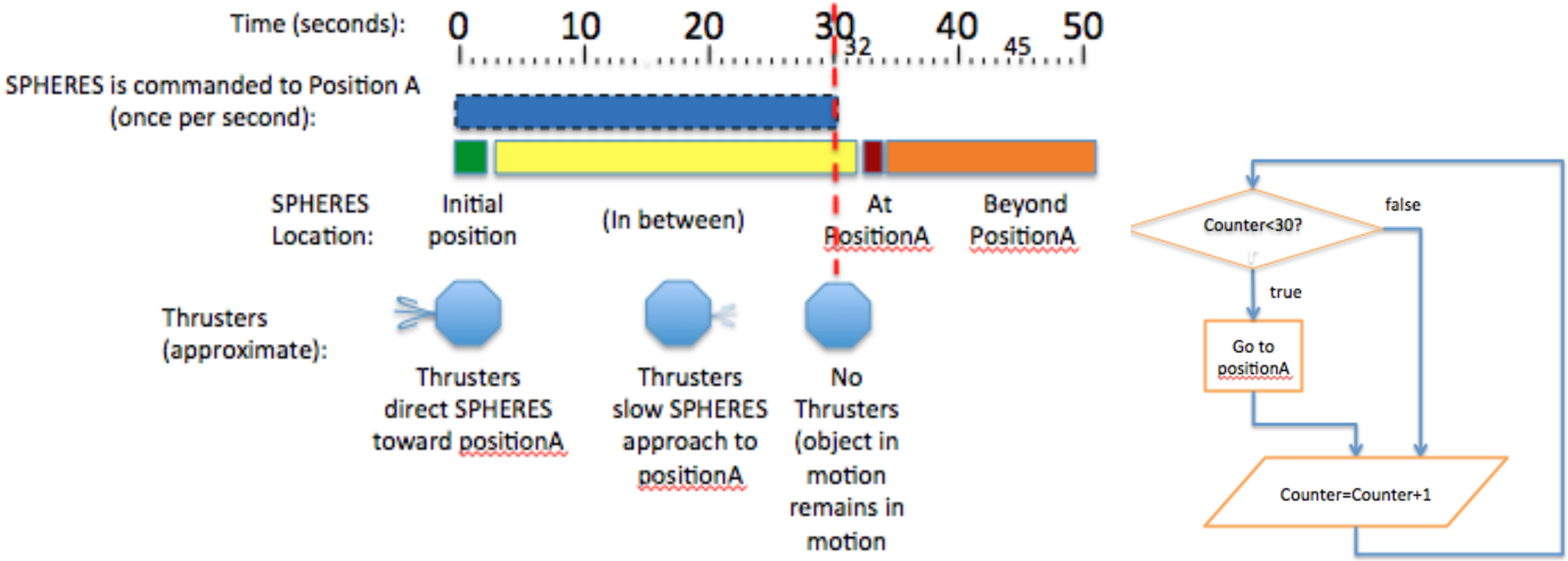


```
loop
    if counter < 30
    then set PositionTarget positionA
    counter = counter + 1
```



# What–if? #1 explained, continued

- What-if? #1 is depicted in the figure below.
  - At 30 seconds:
    - the SPHERES has begun to slow down as it approaches position A
    - the SPHERES is no longer commanded to go to position A



Time (seconds): 0 10 20 30 32 40 45 50

SPHERES is commanded to Position A (once per second):

SPHERES Location: Initial position | (In between) | At PositionA | Beyond PositionA

Thrusters (approximate):

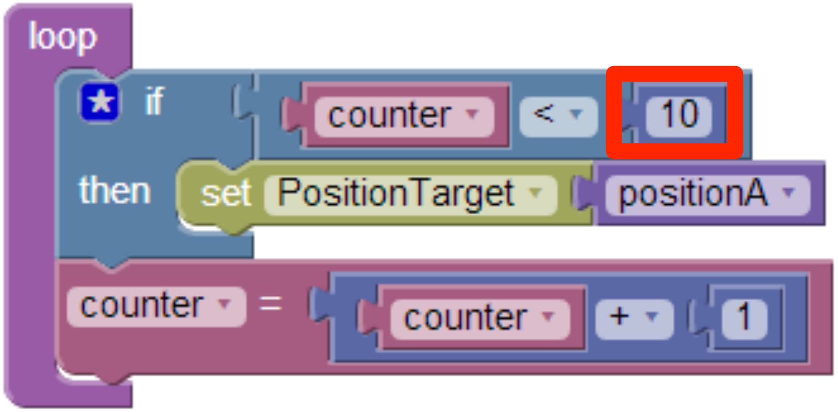
Thrusters direct SPHERES toward positionA

Thrusters slow SPHERES approach to positionA

No Thrusters (object in motion remains in motion

# What–if? #2

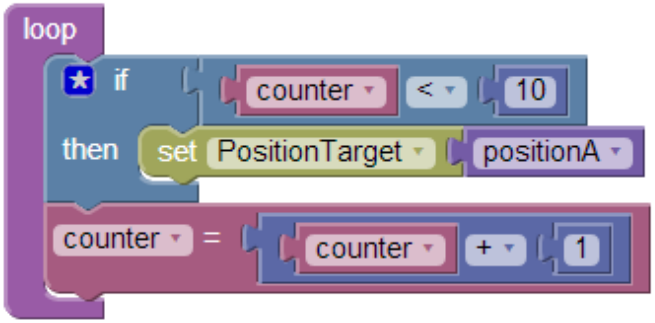
- What if we set counter<10?
- Based on “What-If? #1”, we already know that the conditional statement will not  be true for enough time to allow the SPHERES to reach positionA
- The thrusters will be shut off even sooner than before
- Test your program to see what happens!
  - Compile, Simulate
    - Maximum Time: 90 seconds
  - View simulation at 2x speed



```
loop
    if counter < 10
    then set PositionTarget positionA
    counter = counter + 1
```

# What–if? #2 explained

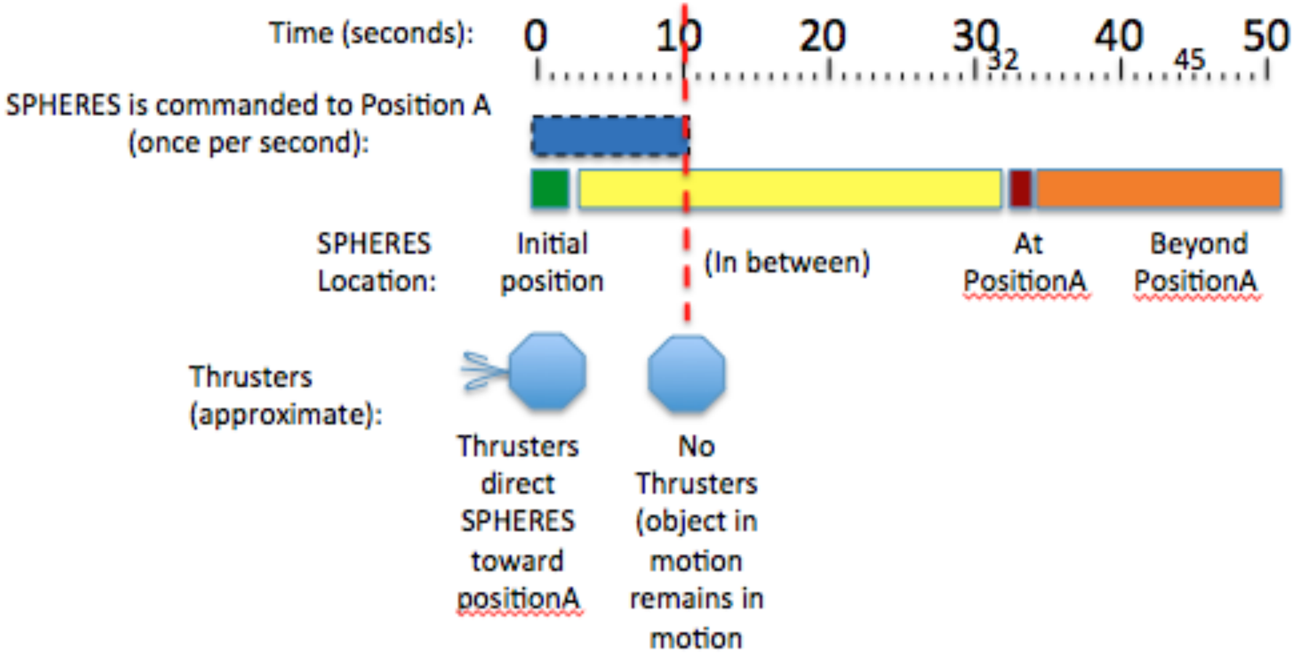
- Notice that this time the SPHERES zips right past point (-1,0,0)
- What happened?
- Again you can explain what happened using Newton's laws
  - This time the SPHERES was moving at a much faster velocity when the thrusters were shut off!!
  - The SPHERES was far enough away from positionA that it hadn't started to slow down yet.
  - "An object in motion remains in motion unless acted on by a force"
  - The SPHERES continued moving at the same velocity it had after the thrusters were shut off



```
loop
  if counter < 10
  then set PositionTarget positionA
  counter = counter + 1
```

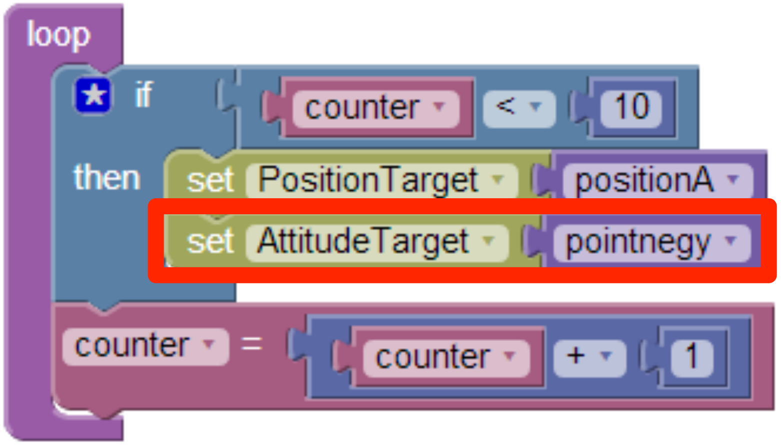
# What–if? #2 explained, continued

- What-if? #2 is depicted in the figure below.
  - At 10 seconds
    - the SPHERES has **not** started to slow down to approach positionA, so it is moving at a faster speed than in what-if? #1
    - the SPHERES is no longer commanded to position A


Time (seconds):
0
10
20
30
32
40
45
50
SPHERES is commanded to Position A (once per second):
SPHERES Location:
Initial position
(In between)
At PositionA
Beyond PositionA
Thrusters (approximate):
Thrusters direct SPHERES toward positionA
No Thrusters (object in motion remains in motion

# What-if? #3

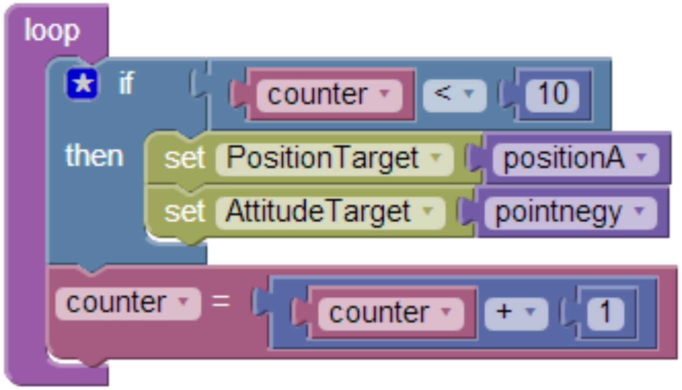
- What if we add a command to change the SPHERES attitude?
- Modify your program as follows:
  - Create the new array
    - `float pointnegy[3]`
      - Set initial value to (0,-1,0)
  - Drag a `setAttitudeTarget` block into the loop after the `setPositionTarget` block
  - Set the `setAttitudeTarget` block to `pointnegy`
- Test your program to see what happens!
  - Compile, Simulate
    - Maximum time: 90s
  - View simulation at 2x speed



```
loop
  if counter < 10
  then set PositionTarget positionA
       set AttitudeTarget pointnegy
  counter = counter + 1
```

# What-if? #3 explained

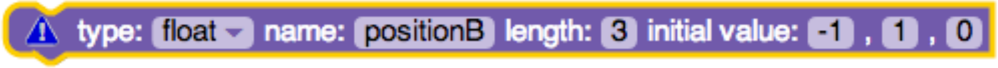
- Notice that this time the SPHERES is tumbling as it zips right past point (-1,0,0)
- What happened?
- Again you can explain what happened using Newton's laws
  - The conditional statement (counter<10) was no longer true ***before:***
    - The SPHERES finished rotating to point toward pointnegy
    - The SPHERES was able to reach positionA.
  - "An object in motion remains in motion unless acted on by a force"
  - The SPHERES was rotating when the thrusters were shut off, so it continued to rotate at the same angular velocity!!



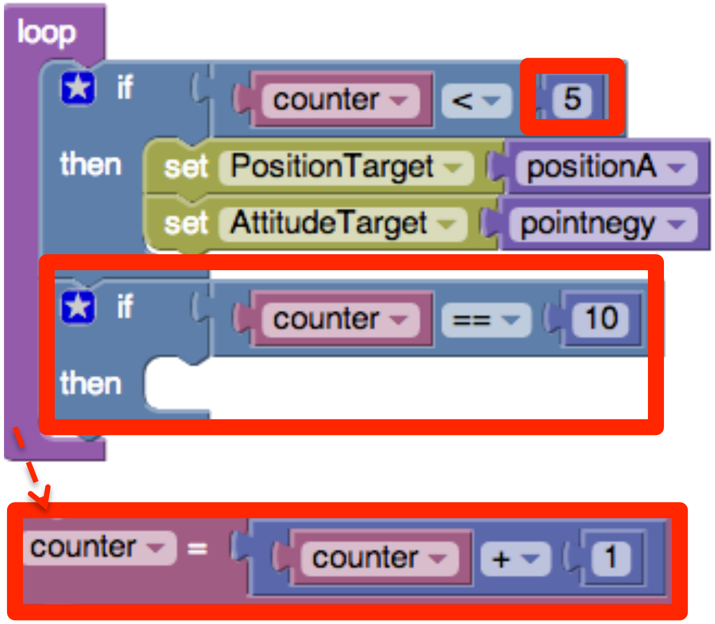
```
loop
  if      counter < 10
  then    set PositionTarget positionA
          set AttitudeTarget pointnegy
  counter = counter + 1
```

# What-if? #4

- What if we add a second “If-then” block with a new position target?
- Modify your program as follows:
  - Create the new array in the “init” page
    - `float positionB[3]`
      - Set initial value to (**-1,1,0**)
  - On the “main” page: Drag the `counter=counter+1` block out of the loop, **but do not delete!**
  - Change the counter in the first “`if-then`” block to 5.
  - Drag an “`If-then`” statement into the loop from the logic accordion
    - Drag” **==**“ from the logic accordion and set it to “**>**”
    - Drag `counter` from variable accordion and a number from the math accordion (10)



type: float name: positionB length: 3 initial value: -1 , 1 , 0

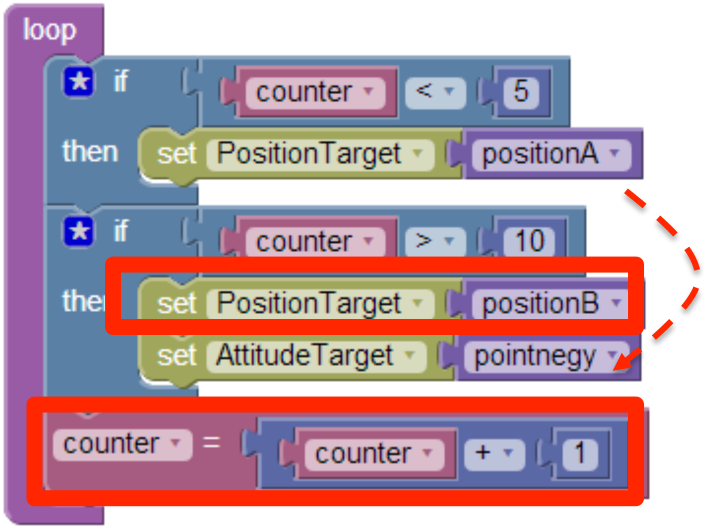


```
loop
  if counter < 5
  then set PositionTarget positionA
       set AttitudeTarget pointnegy
  if counter == 10
  then

counter = counter + 1
```

# What-if? #4, continued

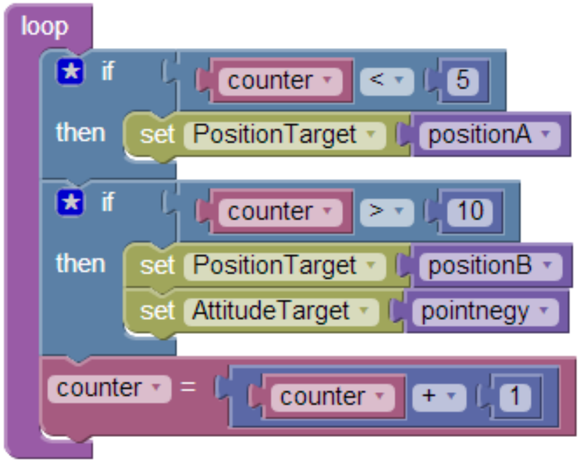
- Modify your program, continued:
  - Drag a setPostionTarget block from the SPHERES Controls accordion into the second “If—then” block ( be sure block is set to “positionB”)
  - Drag the `setAttitudeTarget` block out of the first “`If-then`” block and into the second “`If-then`” block
  - Drag the `counter=counter+1` block back into the loop below the second “If-then” block.
- Test your program to see what happens!
  - Compile, Simulate
    - Maximum Time: 90 seconds
  - **Click the “zoom out” tool** at the bottom of the simulation window to see the end of the simulation
  - View simulation at 2x speed



```
loop
  if counter < 5
  then set PositionTarget positionA
  if counter > 10
  then set PositionTarget positionB
       set AttitudeTarget pointnegy
  counter = counter + 1
```

# What-if? #4 explained

- What did you observe?
  - The satellite started for  positionA but before reaching positionA it swerved off to head for positionB
  - Both the position and the attitude were stable at the end
- Why?
  - The first conditional statement (counter<5) was no longer true ***before*** the satellite was able to reach positionA.
  - The satellite swerved when the second conditional statement(counter>10) was applied
  - The second conditional statement (counter>10) is always true after counter>10 so the program continued to command the satellite to the desired position and attitude



```
loop
  if    counter < 5
  then  set PositionTarget positionA
  if    counter > 10
  then  set PositionTarget positionB
        set AttitudeTarget pointnegy
  counter = counter + 1
```

# Review

- Congratulations! You now have a better understanding of SPHERES dynamics and Newton's first law !

- If you have unexpected results from your own programs, look carefully at how the SPHERES control functions are commanded in your loop.