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

• 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
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 “Text Editor”
  – Create the following variables and arrays: (for help use variables and arrays tutorial)
    • float firstposition[3]
      – Set initial value to (-1,0,0)
    • int counter
      – counter = 0
  – Complete program as shown.

```cpp
void loop()
{
  if (counter<45) {
    api.setPositionTarget(firstposition);
  }
  counter++;
}
```
Expected Dynamics

- Test your program!
  - Compile, Simulate
    - Load settings: Tutorial _90
  - View simulation at 2x speed

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

```c
void loop()
{
    if (counter<45)
    {
        api.setPositionTarget(firstposition);
    }
    counter ++;
}
```
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 position A near time = 32 seconds and stays at position A, even after the counter reaches 45

Next we will try some “what-ifs”.....
What—if? #1

• What if we set counter<30 (instead of <45)?

• Test your program!
  – Compile, Simulate
    • Load settings: Tutorial _90
  – 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 “firstposition” (-1,0,0) the conditional statement (counter<30) was no longer true (see image)

```c
void loop()
{
  if (counter<30) {
    api.setPositionTarget(firstposition);
  }
  counter++;
}
```
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!!

```c
void loop()
{
  if (counter<30)
  {
    api.setPositionTarget(firstposition);
  }
  counter ++;
}
```
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
• 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 firstposition
• The thrusters will be shut off even sooner than before
• Test your program to see what happens!
  – Compile, Simulate
    • Load settings: Tutorial _90
  – View simulation at 2x speed

```cpp
void loop()
{
  if (counter < 10) {
    api.setPositionTarget(firstposition);
  }
  counter++;  
} 
```
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 much faster when the thrusters were shut off.
  – The SPHERES was far enough away from firstposition 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

```java
void loop()
{
  if (counter<10)
  {
    api.setPositionTarget(firstposition);
  }
  counter++;
}
```
• What-if? #2 is depicted in the figure below.
  - At 10 seconds
    • the SPHERES has not started to slow down to approach position A, so it is moving at a faster speed than in what-if? #1
    • the SPHERES is no longer commanded to position A
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
- See what happens!
  - Compile, Simulate
    - Load settings: Tutorial _90
  - View simulation at 2x speed

```cpp
void loop()
{
  if (counter<10) {
    api.setPositionTarget(firstposition);
    api.setAttitudeTarget(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 negy
    - The SPHERES was able to reach firstposition.
  
  - “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.

```java
void loop()
{
  if (counter<10) {
    api.setPositionTarget(first position);
    api.setAttitudeTarget(pointnegy);
  } 
  counter++;
}
```
What-if? #4

- What if we add a second “If statement” with a new position target?

- Modify your program as follows:
  - Create the new array
    - float secondposition[3]
      - Set initial value to (-1,1,0)
  - Add the second if statement as shown (note the > symbol).
  - The counter in the first if statement to counter <5 changed.

```c
void loop()
{
  if (counter<5) {
    api.setPositionTarget(firstposition);
    api.setAttitudeTarget(pointnegy);
  }
  if (counter>10) {
    api.setPositionTarget(secondposition);
  }
  counter++;
}
```
What-if? #4, continued

- Move the `api.setAttitudeTarget` statement from the first if statement to the second if statement as shown.

- Test your program to see what happens!
  - Compile, Simulate
    - Load settings: Tutorial _90
  - **Click the “zoom out” tool** at the bottom of the simulation window to see the end of the simulation
  - View simulation at 2x speed

```cpp
void loop()
{
    if (counter<5) {
        api.setPositionTarget(firstposition);
    }
    if (counter>10) {
        api.setPositionTarget(secondposition);
        api.setAttitudeTarget(pointnegy);
    }
    counter++;
}
```
What-if? #4 explained

• What did you observe?
  – The satellite started for firstposition but before reaching first position it swerved off to head for secondposition
  – 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 firstposition.
  – 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

```cpp
void loop()
{
  if (counter<5) {
    api.setPositionTarget(firstposition);
  }
  if (counter>10) {
    api.setPositionTarget(secondposition);
    api.setAttitudeTarget(pointnegy);
  }
  counter++;
}
```
• 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.