ZERO ROBOTICS

ISS PROGRAMING CHALLENGE

Game Overview

2014 Middle School Program

V1.2

DARPA NASA CASIS MAP Massachusetts Afterschool Partnership EDC Learning transforms lives. [TopCoder] Aurora Flight Sciences MIT SSL MIT

Goals

- Part 1: Game Overview
  - To be used at the end of Week 1
  - Introduces the game in preparation for the “Acting out the Game” activity during Field Day
  - Additional details are available in the game manual

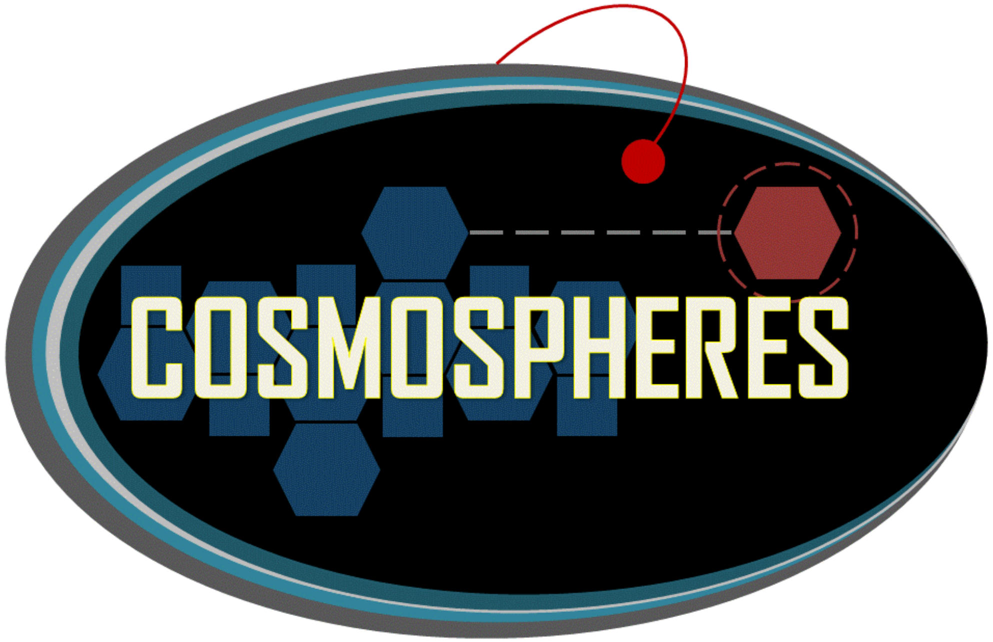
- Part 2: Game Specific Functions Tutorial
  - To be used at the end of Week 2
  - Provides practice exercises for using some of the Game Specific functions

ZERO ROBOTICS
ISS PROGRAMING CHALLENGE

Your Mission

- **Critical Operation to Save Mankind from Obliteration (COSMO)**
  - Your Mission: Use a robotic satellite to change the path of an incoming comet headed toward your home base
  - Your Goal: Deflect the path of the comet the furthest from your home base

- Methods you may use:
  - Gravitational Attraction
  - Laser Pellet Repulsion



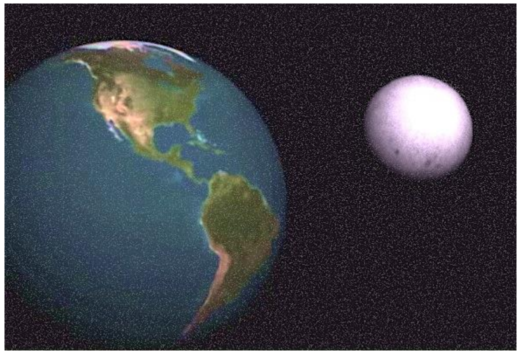
What is Gravitational Attraction?

And how does…:

- Increasing the mass of the SPHERE (by collecting debris)
- Decreasing the distance between the SPHERE and the comet (by moving the SPHERE closer to the comet)

…affect the force of gravitational attraction acting on the comet?

How does mass affect this force?



How does distance affect this force?

•Watch for answers in the following short video clips:

- https://www.youtube.com/watch?v=xICEt51A-Ac
- https://www.youtube.com/watch?v=jubO40j-rXc&feature=youtu.be

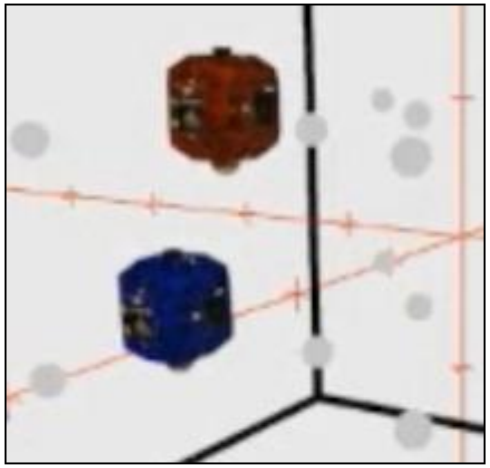
Notes from 2nd video:
•A Newton is a standardized unit of force;
•1.45×10^{-7} Newtons = 0.000000145 Newtons

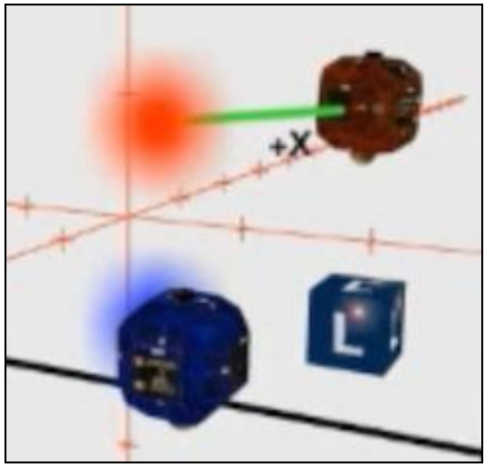
ZERO ROBOTICS
ISS PROGRAMING CHALLENGE

Your Tasks

Your tasks:

- Phase 1
  - Navigate debris field
  - Collect debris, if desired to increase mass, using either:
    - Collision
    - Lasso
  - Collect laser pack(s), if desired
- Phase 2
  - Deflect path of comet away from your home base by using:
    - gravitational attraction,
    - laser repulsion
    - both





Game Layout – Phase 1

- Phase 1 has two zones:

  Zone 1: Debris field

  Zone 2: Laser packs

  - x, y coordinates for zone boundaries are shown
  - SPHERES can only move in the x,y plane

Interaction Zone Dimensions

X [m]	-0.64 : +0.64
Y [m]	-0.80 :+0.80
Z[m]	±0.2

SPHERES Satellite Starting Position

X [m]	0.0
Y [m]	-0.80
Z[m]	±0.2

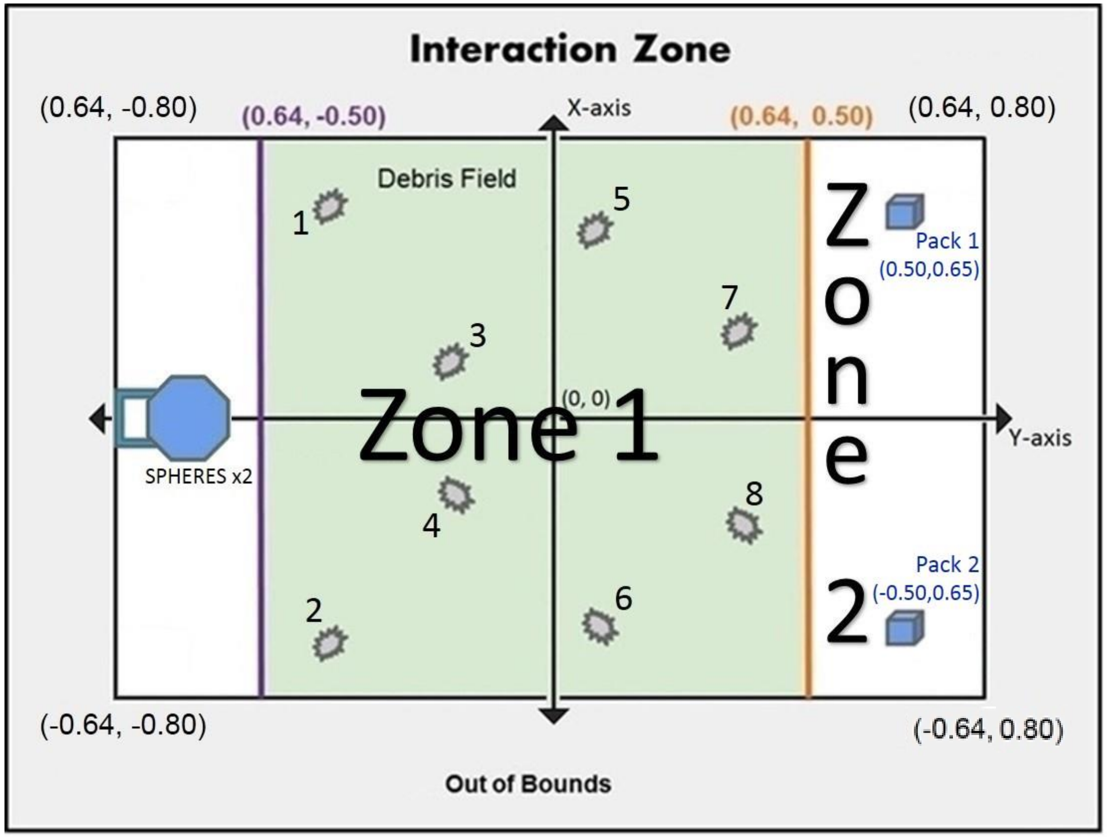


Diagram not to scale

Notes:

•SPHERES are automatically constrained to separate z planes (+/– 0.2 as shown) to avoid interference during the game.

• It is appropriate to use z=0 for all z coordinates in your programs

Game Layout – Phase 2

Phase 2:

- Begins 90 seconds after start of game
- Zone 3
  - Comet Entry Point
  - Home Base
  - x, y coordinates for Zone 3 features are shown

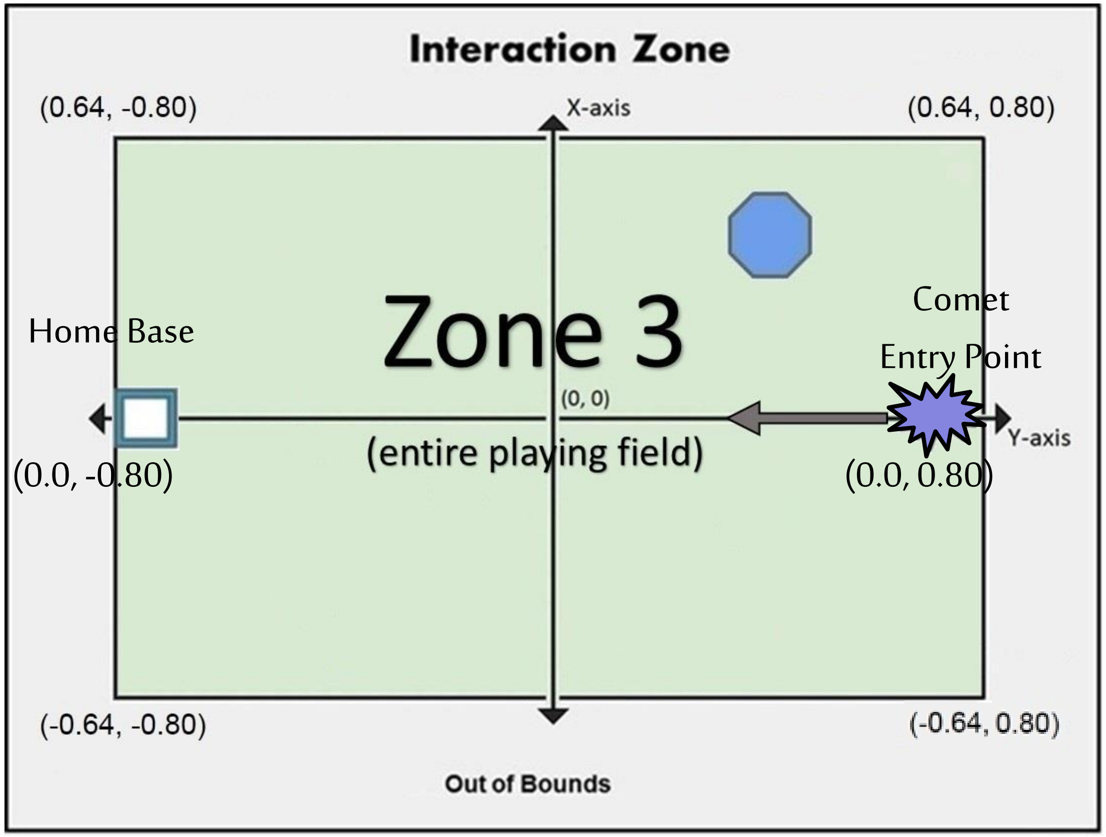


Diagram not to scale

Zone 1– Debris locations

- Each player's Zone 1 contains eight identical pieces of debris

- x,y coordinates for debris locations:

Debris Number	Location
1	(0.523, -0.5)
2	(-0.523, -0.5)
3	(0.18, -0.2)
4	(-0.18, -0.2)
5	(0.43, 0.15)
6	(-0.43, 0.15)
7	(0.15, 0.37)
8	(-0.15, 0.37)

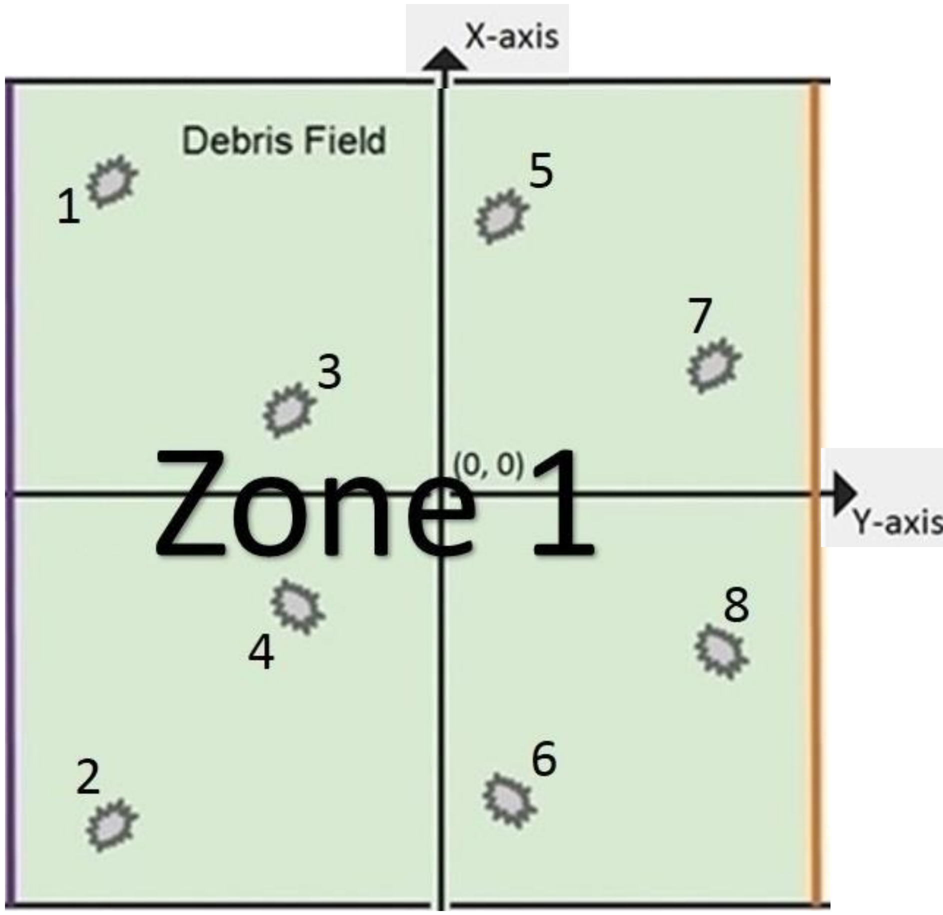
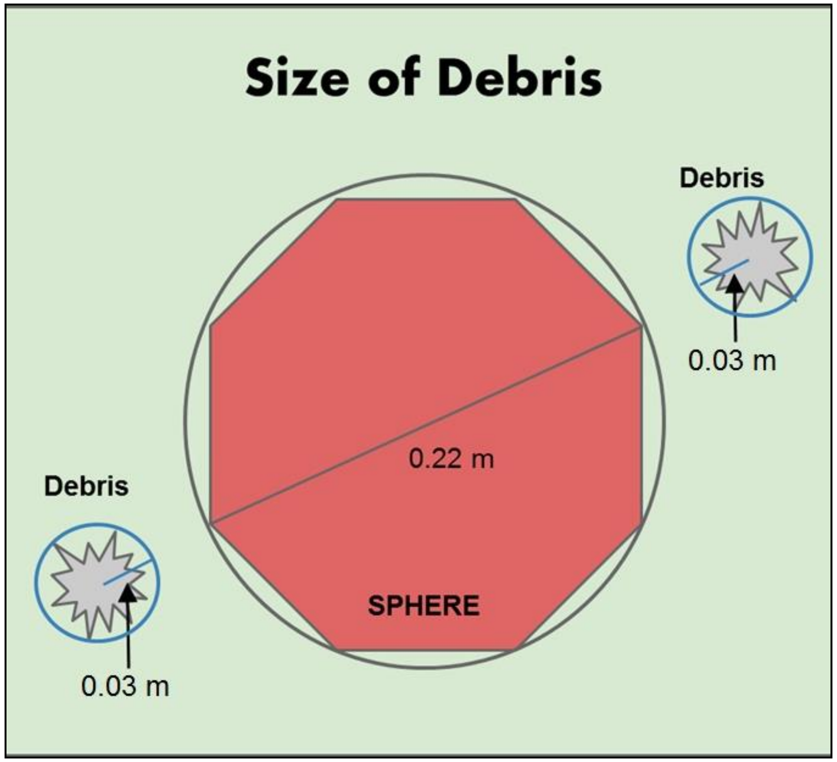


Diagram not to scale

ZERO ROBOTICS
ISS PROGRAMING CHALLENGE

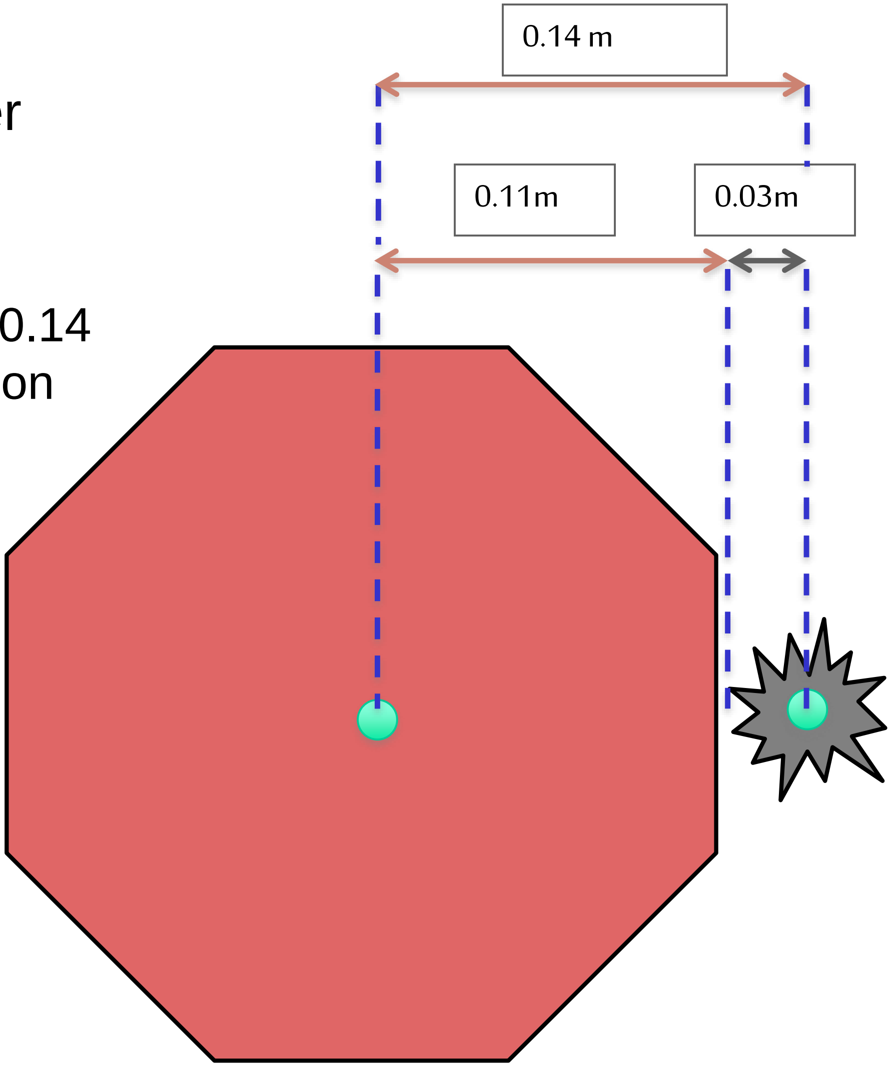
Zone 1 – Debris Field

- You may choose to:
  - Avoid debris
    - Find a path through the debris field that a SPHERES satellite can navigate without colliding with debris
  - Pick up debris to increase your mass
    - You may pick up as many as all 8 debris
    - Two ways to pick up debris
      - Collision (easier)
      - Lasso (harder)
        - » Collects more mass with fewer fuel penalties



Zone 1 –Avoiding Debris

- The locations of the SPHERE and debris are described by their center points
- To avoid collision with debris:
  - The SPHERE must be **more** than 0.14 meters away from the debris location

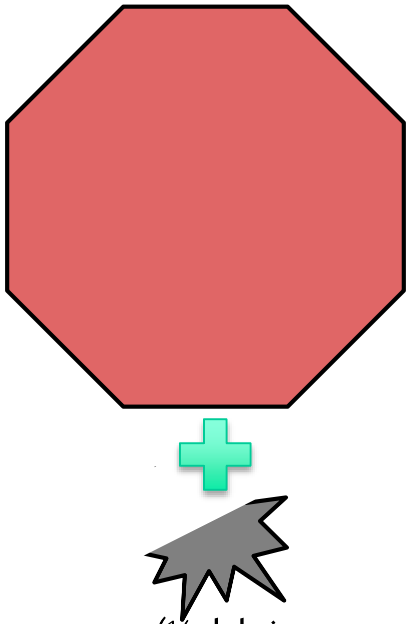


Debris and SPHERES Measurements

Debris radius (m)	0.03
SPHERES radius (m)	0.11

Zone 1 –Collect Debris by Collision

- To collect debris by collision:
  - Find the location of the debris
  - Command SPHERES to move to collide with debris
  - Half of the debris mass is added to your SPHERES mass
    - Some of the debris mass is lost during collision
- What happens when you collide with debris?
  - SPHERES immediately slows down (translational and rotational)
  - Player controls lock for 3 seconds
    - Normal control returns after 3 seconds
  - Fuel is used to slow the satellite during a collision
    - Amount of fuel used depends on speed of collision

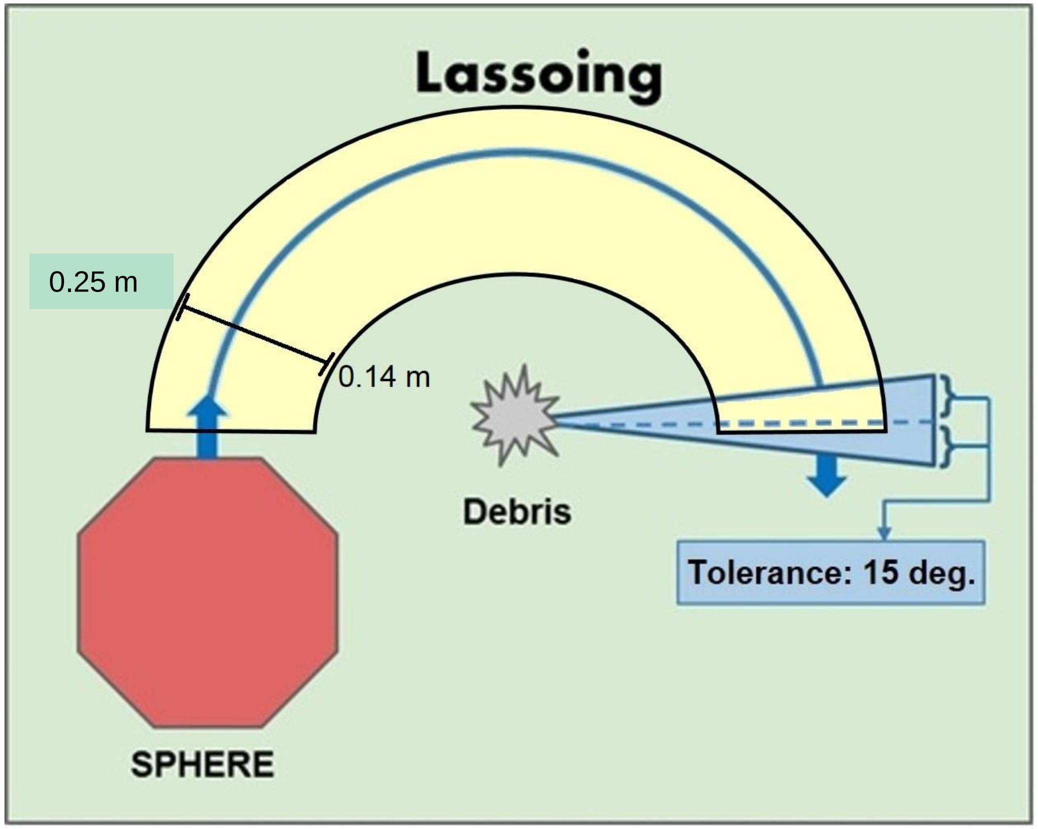


Debris and SPHERES initial mass

Debris mass	4.5×10^{-6}
Initial SPHERES mass	1.0×10^{-5}

Zone 1 – Collect Debris by Lasso

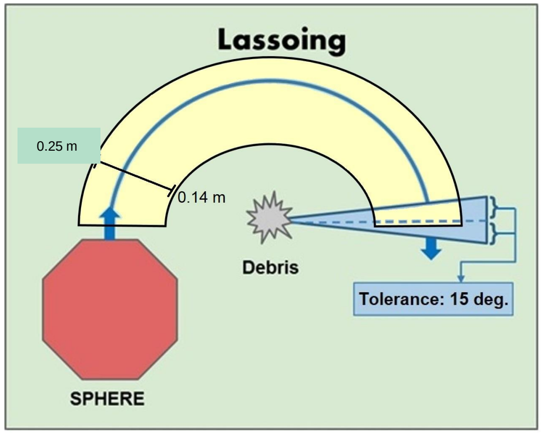
- How to Lasso? (part 1):
  - SPHERES must travel from one side of the debris to the other, in an approximately semicircular path
  - The distance between the center of the SPHERES and the center of the debris must be between 0.14 m and 0.25 m at all times
  - If a collision occurs with the debris being lassoed, it will count as a collision, not as lassoing

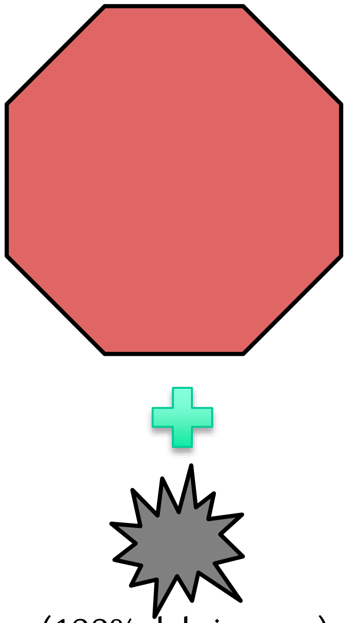


Zone 1 – Collect Debris by Lasso, cont.

How to lasso (part 2):

- Use (game-specific functions described later):
  - startLasso: to initialize the lassoing process
  - setPositionTarget: used multiple times to stay inside the semi-circular path
- 100% of the mass of the debris will be added to your SPHERES
- No fuel penalty

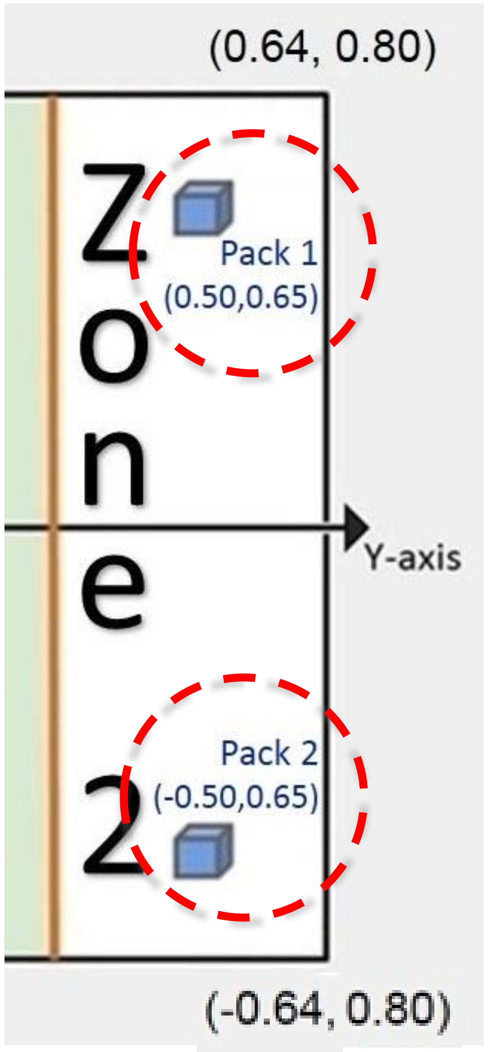



(100% debris mass)

Zone 2 – Laser Packs

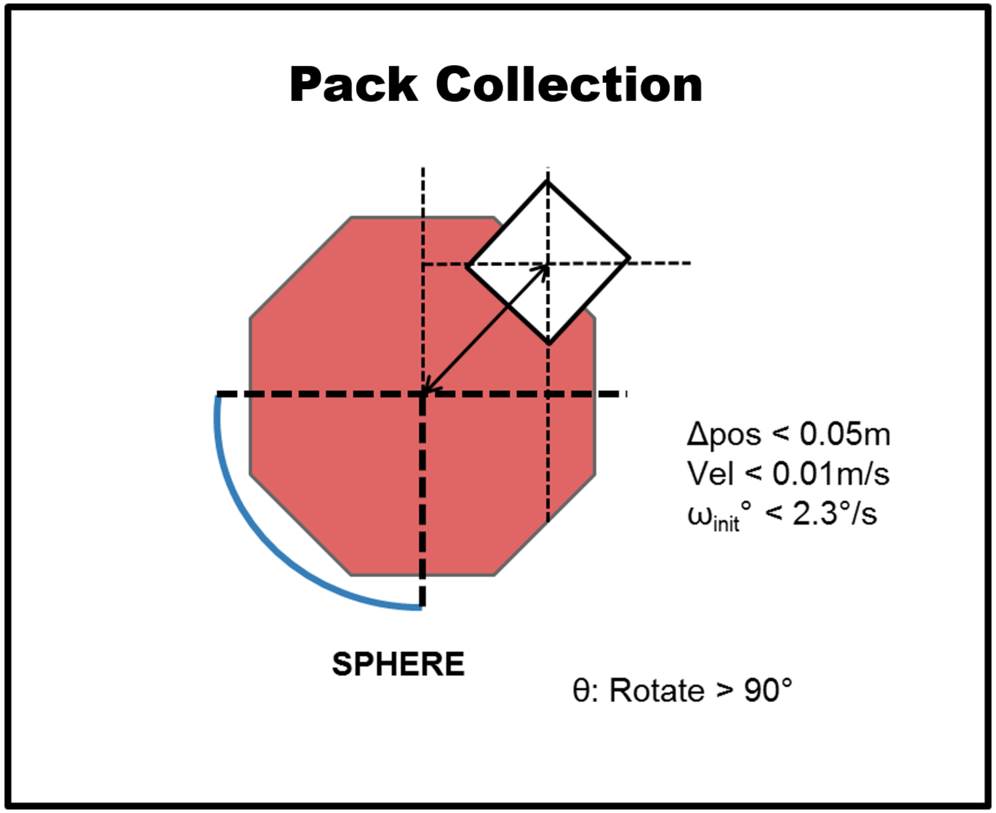
- You may choose to pick up 0,1, or 2 laser packs
- Laser packs are identical and have 10 shots each.
- The locations of the laser packs are shown on the diagram.
- Laser packs may be picked up by either SPHERE
  - Once the laser pack is picked up by one team it is no longer available for pick up

    by the other team.



Zone 2 – Picking up Laser Packs

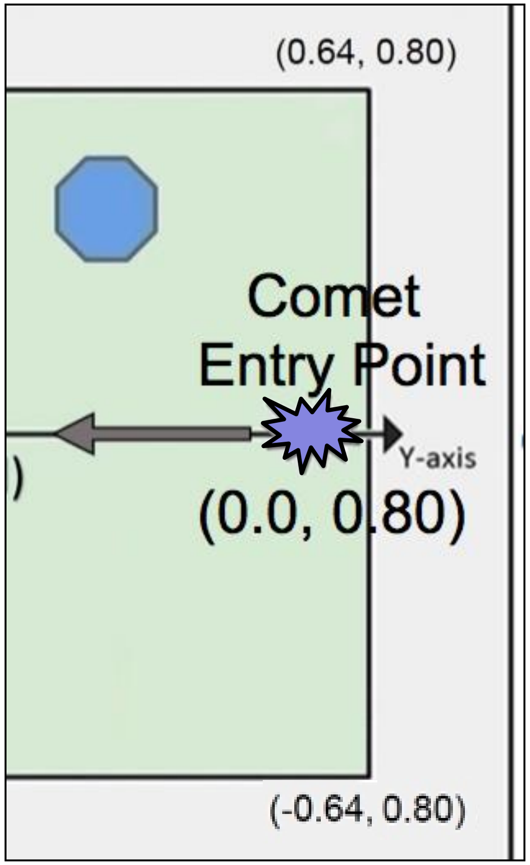
- Locate the laser pack
- Stop the SPHERES
  - within 0.05 m of the location of the pack
- Rotate SPHERES 90˚
  - to pick up the laser pack
- Use game-specific function havePack (described later)
  - to check whether the laser pack has been picked up



Phase 2 –Comets

- When do comets enter?
  - At time = 90 seconds
- Where do comets enter?
  - at x,y coordinates =(0.0, 0.80)
- What is the initial velocity of comet?
  - 0.03 m/s in a straight line towards its SPHERES' home base
- Game-specific functions (described later)
  - predictCometState: predicts where the comet will be at a particular time
  - getCometState: stores the current location of the comet

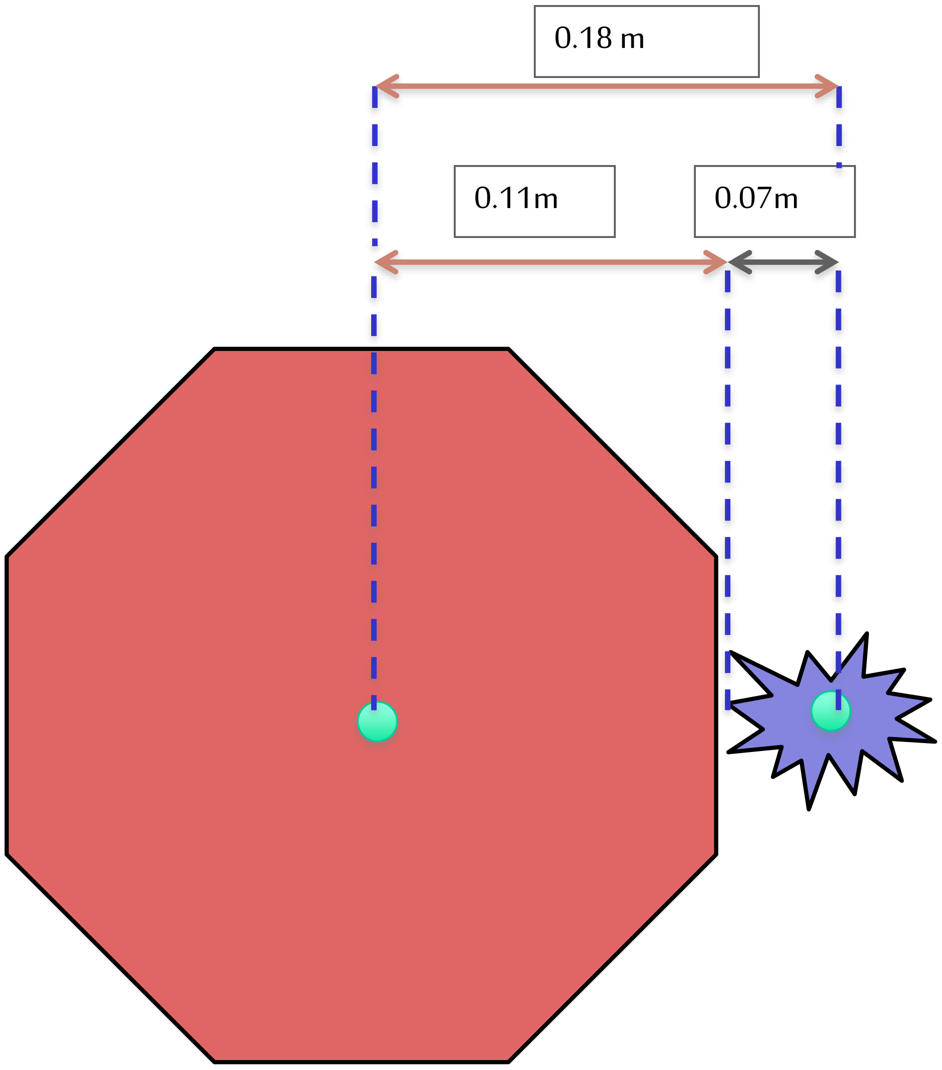
- The goal in Phase 2 is to deflect your comet away from your home base



ZERO ROBOTICS ISS PROGRAMING CHALLENGE

Phase 2 –Comet Collision

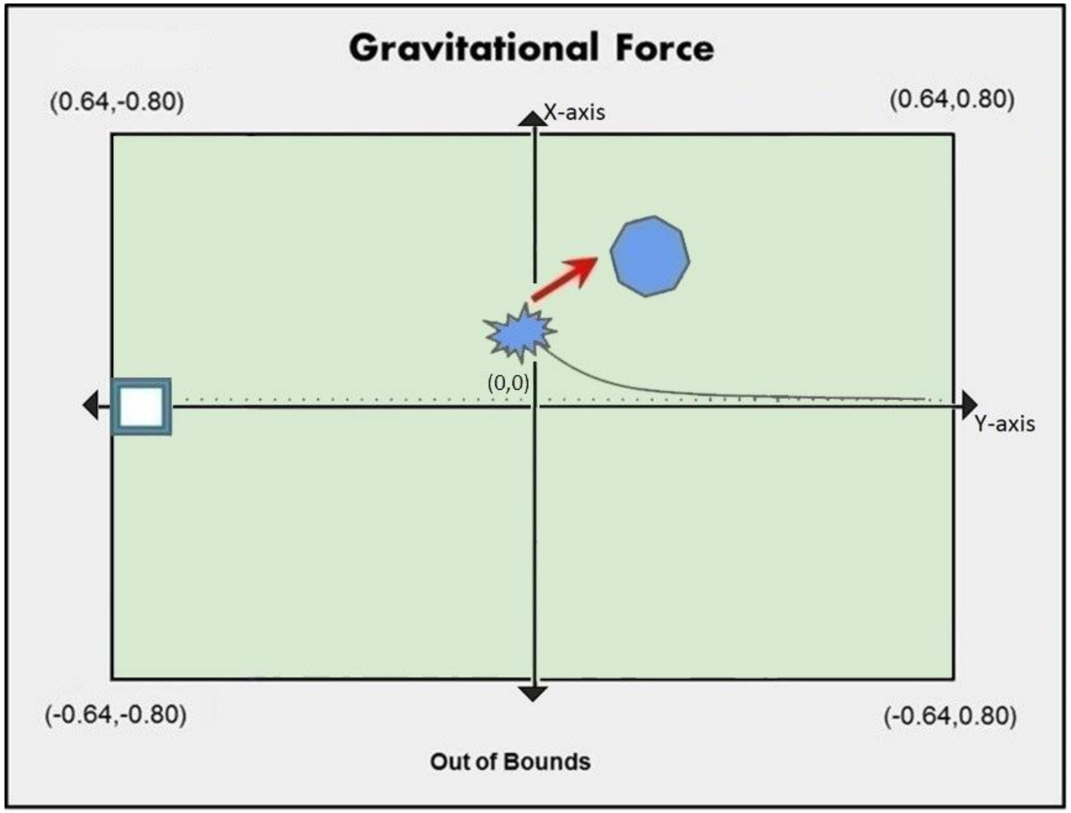
- To avoid collision with the comet:
  - The SPHERE must be **more** than 0.18 meters away from the comet
- If a satellite collides with its comet
  - The satellite will **<u>bounce back</u>** (in the direction from which it came at the same speed with which it hit the comet)
  - Player controls will lock (until the distance between the center of the SPHERES and its comet is greater than 0.18 m.)
    - The game-specific function isBounceActive checks whether the SPHERES is under collision conditions with its comet



DARPA NASA CASIS MAP EDC Learning transforms lives. [TopCoder] Aurora FLIGHT SCIENCES MIT SSL MIT

Phase 2– Deflecting Comet using Gravity

- Remember: you can increase the force of gravitational attraction on the comet by:
  - **Phase 1:** Increasing the mass of the SPHERE (by collecting debris)
  - **Phase 2:** Decreasing the distance between the SPHERE and the comet (by moving the SPHERE closer to the comet)



- Notes: for CosmoSPHERES:
  - G is normalized to 1
  - $m_{comet} = 1$

Diagram not to scale

Effect of change is exaggerated

$$F_{gravity} = \frac{G(m_1)(m_2)}{distance^2} = \frac{(1.0\text{x}10^{-5}{}_{\text{simulated mass of SPHERE}} + m_{\text{collected debris}})(m_{comet})}{distance^2}$$

Phase 2– Deflecting Comet using Lasers

- How does the laser work?
  - The laser changes the comet's velocity when it hits the comet
    - Notes: Each laser impulse changes the comet velocity by 0.00115 m/s (small change)
- How fast can the laser be fired?
  - Once per second
- What is the laser's range (how far can it shoot)?
  - You can fire from any range
- Can I shoot my opponent's comet?
  - The laser will not affect the opponent's comet or SPHERES.

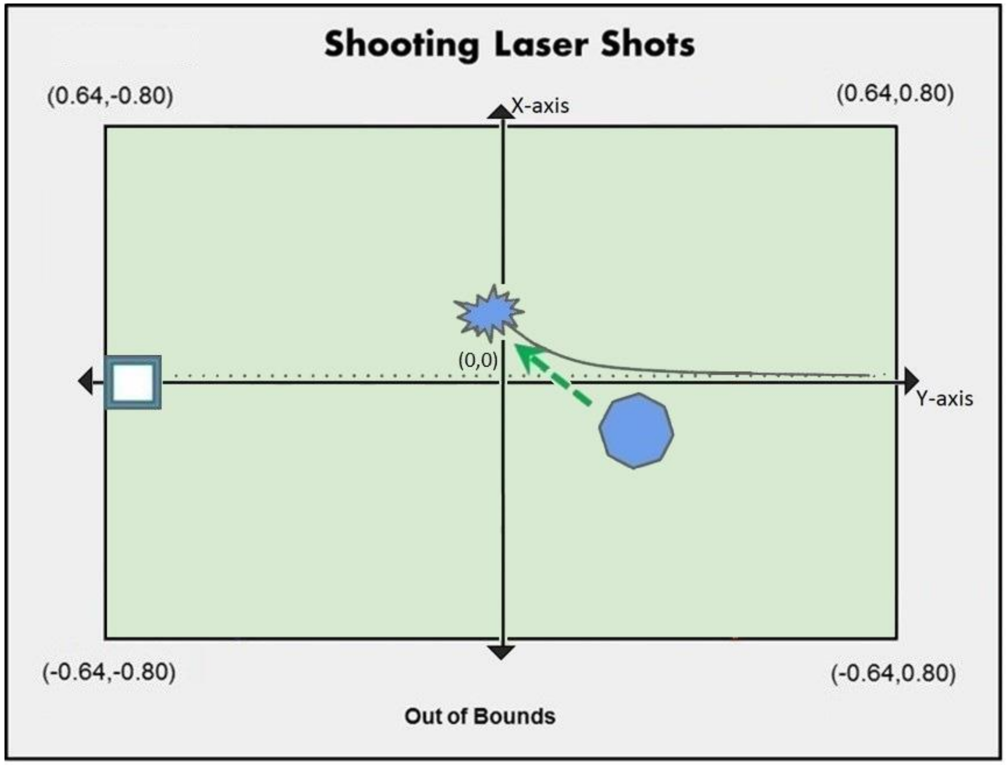


Diagram not to scale

Effect of laser is exaggerated

Phase 2– Deflecting Comet using Lasers

- How do I fire the laser?
  - Use the following game-specific functions (described in more detail later):
    - "faceTarget": rotates the satellite to face the comet
    - "isFacingComet": checks if the satellite is facing the comet
    - "shootLaser" shoots a laser at the comet
    - "laserShotsRemaining": checks the number of laser shots you have left
  - Comet's path is changed if the laser beam touches any part of the comet
    - Note: All hits are treated like direct hits through the center of the comet
  - If the SPHERES is not facing the comet when it shoots, the laser will miss the comet

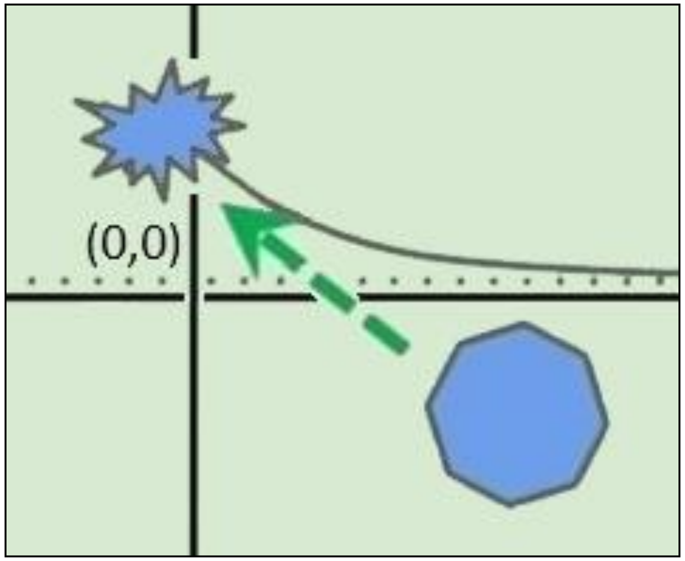
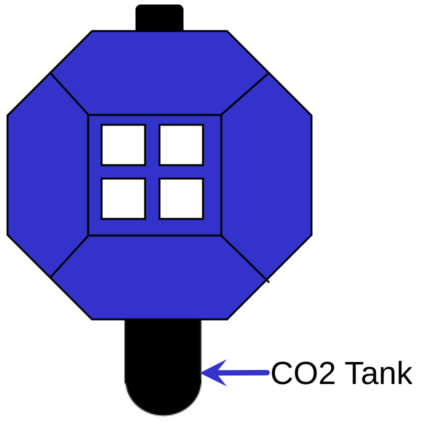


Diagram not to scale

Initial Conditions, Fuel Penalties, and the End of the Game

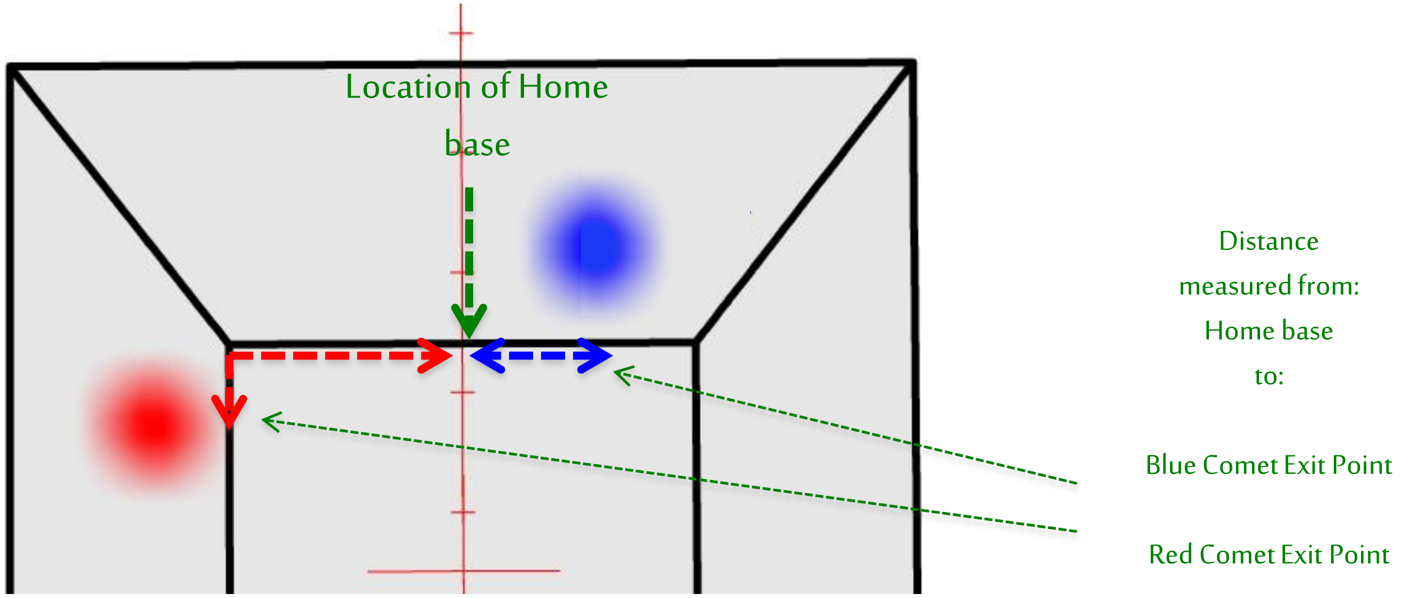
- Initial conditions:
  - Total time: 180 seconds
    - (90 seconds in Phase 1, 90 seconds in Phase 2)
  - Virtual fuel: 50 seconds thruster firing time
- Fuel is used when thrusters are fired:
  - Motion commanded by player
  - SPHERES motion during:
    - Collisions with debris
    - Collision with comet
    - Out of bounds conditions

    Note: Amount used depends on speed of SPHERES at time of event
  - Additional Out-of-bounds penalty
    - Fuel penalty = 2.5 * (seconds out of bounds)
- End of the Game occurs when either :
  - both comets have left the Interaction Zone
  - game is out of time at 180 seconds



Scoring

- Your score is based only on the distance between your home base and where your comet left the Interaction Zone:
  - This distance is measured around the perimeter of the Interaction Zone.
  - The Zero Robotics team uses this distance to calculate your score.
  - In order to maximize your score, your comet should exit the arena as far from your base as possible.



Start to plan your strategy

As you plan your strategy, here are some choices to make:

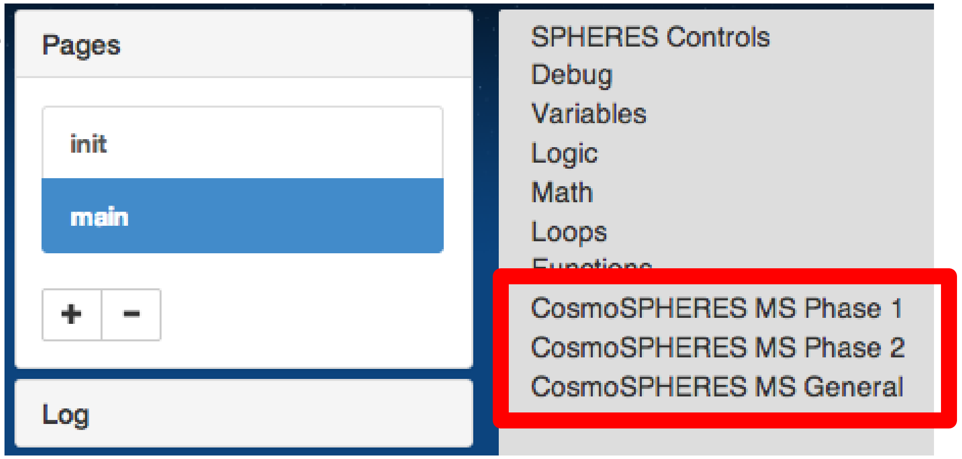
- Do you want to use gravity, lasers, or a combination of the two to deflect the comet?
- Zone 1
  - Do you want to pick up debris? If so, how many?
  - Will you pick up debris by collision, by lasso or use both methods?
  - How will you navigate through the debris field?
- Zone 2
  - Do you want to pick a laser pack(s)?
  - If so, do you want to pick up one or both?
  - If both, which will you pick up first?
- Zone 3
  - Where do you want to position yourself to deflect the comet upon comet entry?
  - Do you want to remain stationary, or do you want to move to follow the comet?

End of Part 1 materials

This slide has been intentionally left blank.

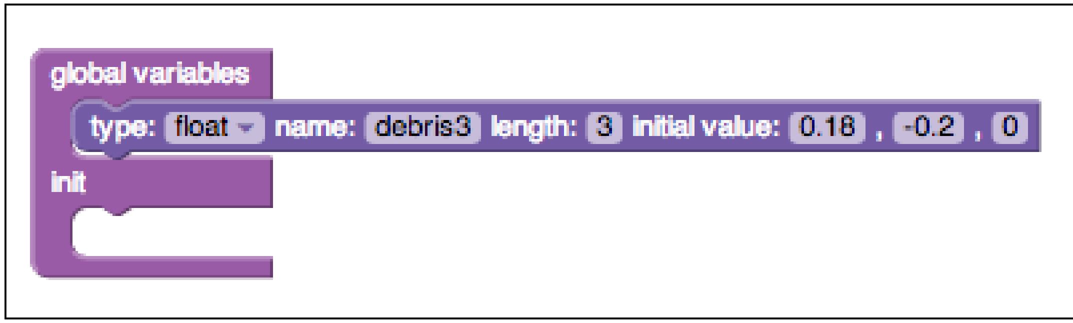
Part 2: Game Specific Functions

- The accordion for the Game Specific functions included in the Graphical Editor IDE for the CosmoSPHERES MS Game are shown here
  - It is not necessary to use all of the game specific functions provided to complete game objectives
- This tutorial will introduce some of the Game Specific functions for
  - CosmoSPHERES MS
    - Phase 1
    - Phase 2
    - Phase 3
- All other Game Specific functions will follow a similar pattern as those introduced

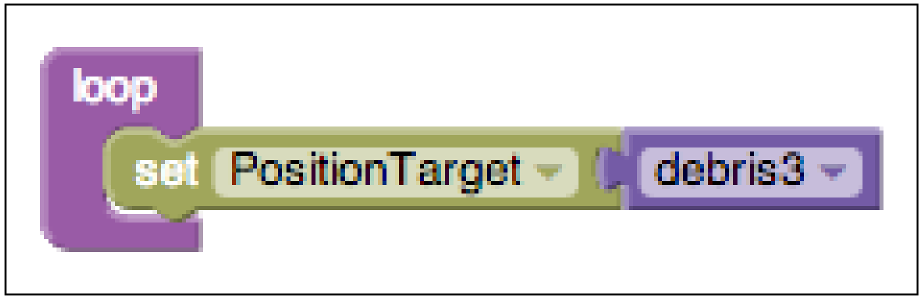


MS2014 Zone 1- Debris Collision

- Write a program to collide with debris 3
  - Located at 0.18,-0.20
- Create a new project using
  - Editor: Graphical Editor
  - Game: CosmoSPHERES_MS
- Declare:
  - Float debris3 [3]
  - Initialize debris3 to (0.18, -0.2, 0)
- Drag a setPositionTarget block into the loop and set toggle to "debris3"



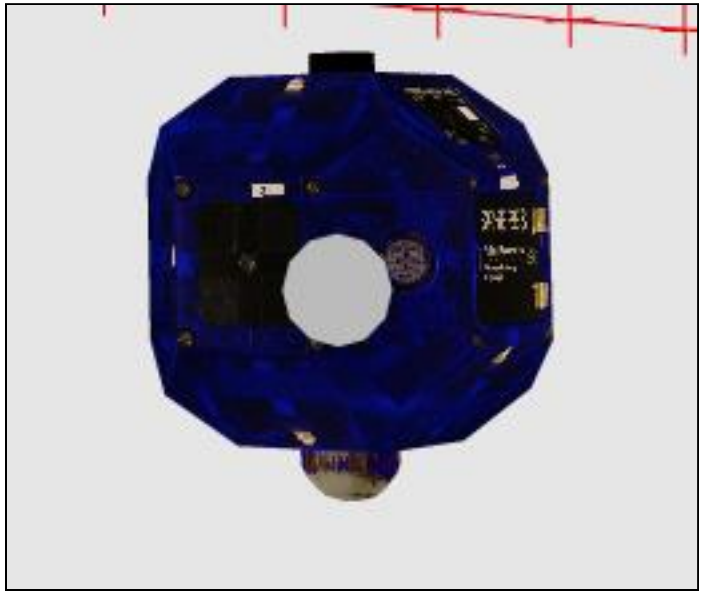
```
global variables
  type: float  name: debris3  length: 3  initial value: 0.18 , -0.2 , 0
init
```



```
loop
  set PositionTarget  debris3
```

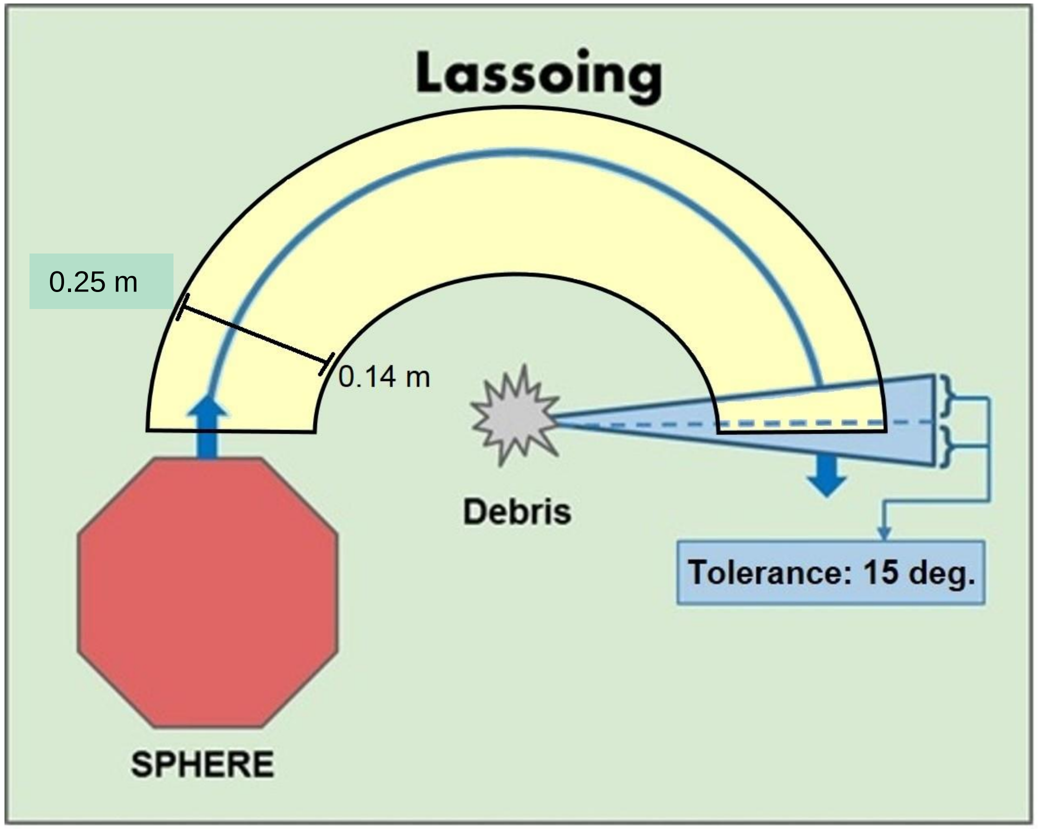
MS2014 Zone 1- Debris Collision

- Simulate and View Results
- You should see the SPHERE pick up debris3



MS2014 Zone 1- Start Lasso

- To lasso debris:
  - Move SPHERE from one side of the debris to the other, in an approximately semicircular path
  - Keep the distance between the center of the SPHERES and the center of the debris between: 0.14 m and 0.25 m
- Things to watch out for
  - Moving Out of Bounds” when lassoing
  - Colliding with debris when lassoing

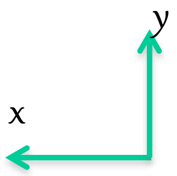


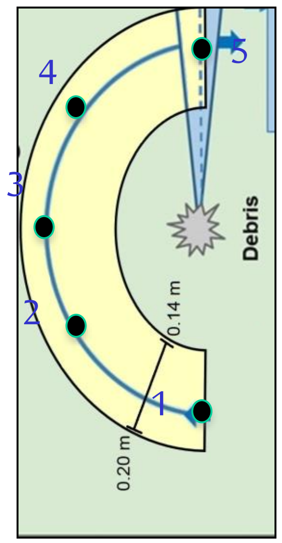
ZERO ROBOTICS ISS PROGRAMING CHALLENGE

MS2014 Zone 1- Start Lasso, Example

- Let’s test on debris 3
  - Located at 0.18,-0.20

- Pick points in the semicircular path around the debris (use graph paper)
  - SPHERES must be between 0.14 and 0.25 meters away from the debris during lasso.
  - Points 1,3,5 shown in the table are picked 0.18 meters from the debris, as an example
  - You may want to try with only 3 points





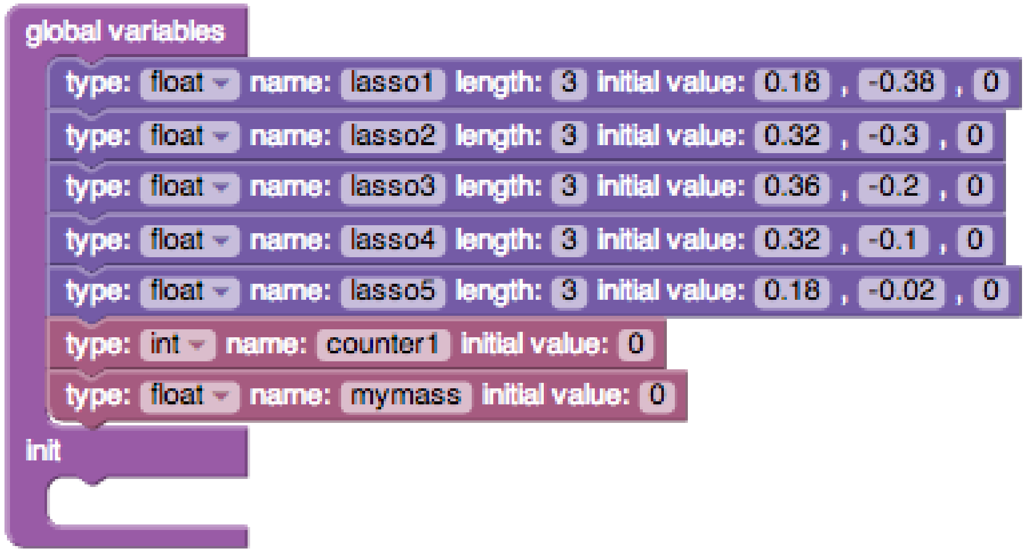
(0.18, -0.2)

point	X (0.18)	Y (-0.2)	
1	0.18	**-0.38**	(y) - (0.18m)
2	*0.32*	*-0.3*	*estimated*
3	**0.36**	-0.2	(x) + (0.18m)
4	*0.32*	*-0.1*	*estimated*
5	0.18	**-0.02**	(y) + (0.18m)

DARPA NASA CASIS MAP EDC Learning transforms lives. [TopCoder] Aurora Flight Sciences MIT SSL MIT

MS2014 Zone 1- Start Lasso, Example

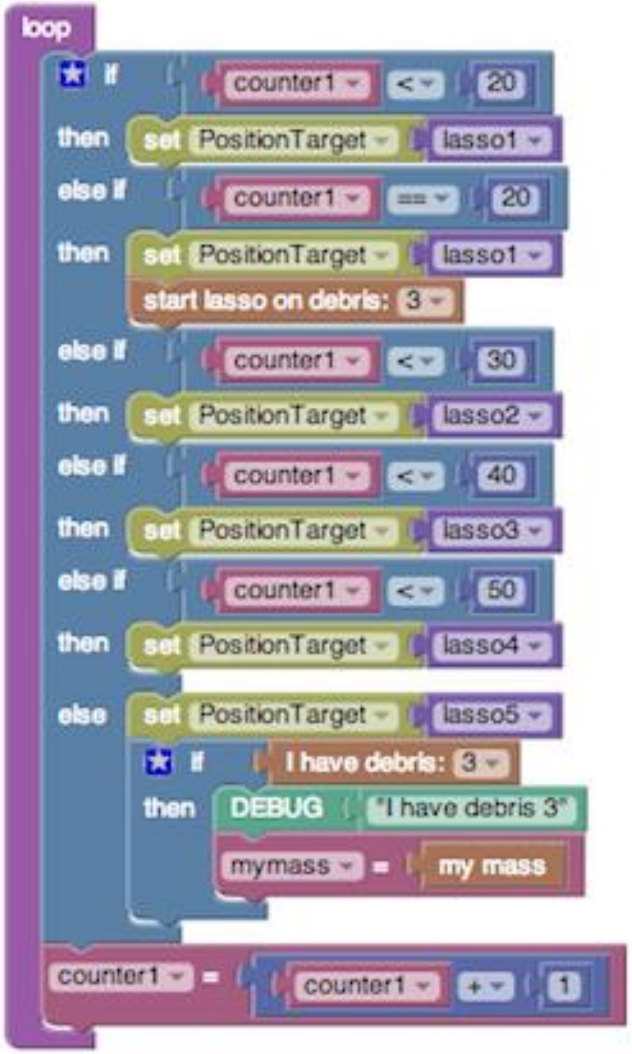
- Create a new project using
  - Game: CosmoSPHERES MS
  - Initial Editor: Graphical Editor
- Declare all Arrays and variables on the Init page
  - Ex:
    - lasso1-lasso5,
    - counter, my mass
- On the main page use conditional statements, and the setPositionTarget to move the SPHERE from point to point within the semicircular path



global variables
type: float name: lasso1 length: 3 initial value: 0.18 , -0.38 , 0
type: float name: lasso2 length: 3 initial value: 0.32 , -0.3 , 0
type: float name: lasso3 length: 3 initial value: 0.36 , -0.2 , 0
type: float name: lasso4 length: 3 initial value: 0.32 , -0.1 , 0
type: float name: lasso5 length: 3 initial value: 0.18 , -0.02 , 0
type: int name: counter1 initial value: 0
type: float name: mymass initial value: 0
init

MS2014 Zone 1- Start Lasso, Example

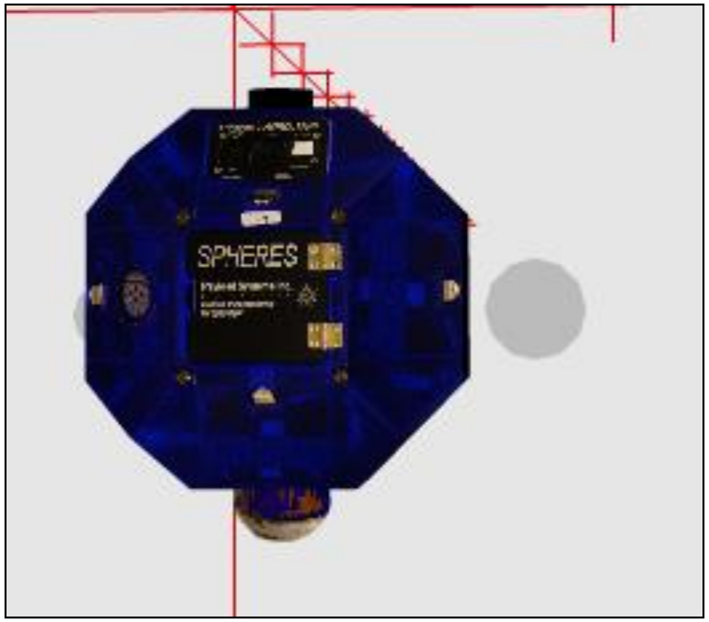
- Call the startLasso function **after** the SPHERE has moved to the first point in the lasso process
- The startLasso function starts Lasso on a piece of debris
    - This function should only be called once per debris at the start of lasso
    - Use == to make sure the startlasso command only happens once
- Use “I have debris” function and add a DEBUG message to confirm pick up as shown
    - Remember to use the quotation marks.



```
loop
  if counter1 < 20
  then set PositionTarget lasso1
  else if counter1 == 20
  then set PositionTarget lasso1
       start lasso on debris: 3
  else if counter1 < 30
  then set PositionTarget lasso2
  else if counter1 < 40
  then set PositionTarget lasso3
  else if counter1 < 50
  then set PositionTarget lasso4
  else set PositionTarget lasso5
       if I have debris: 3
       then DEBUG "I have debris 3"
            mymass = my mass
  counter1 = counter1 + 1
```

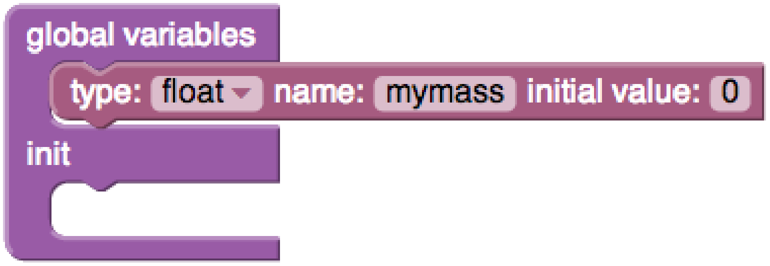
MS2014 Zone 1- Start Lasso, Example

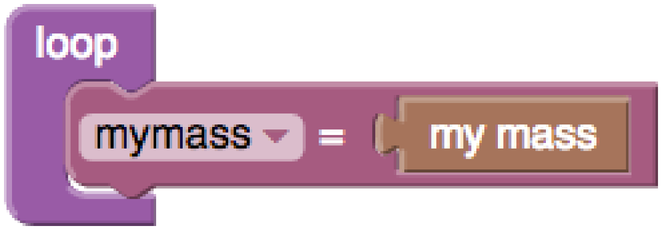
- Simulate and run your program!
- You should see the SPHERE lasso the debris without colliding
- A debug message should appear in the console once per second after the debris has been picked up



MS2014 Zone 1- Check Mass

- One way to check mass:
  - On the init page:
    - Create a variable, ex. mymass, to store the value returned from the function that gets the SPHERES mass
  - On the main page:
    - Assign your variable (ex. mymass) = the value of the block “my mass”
      - Use “Select = 0 “ block
      - Go to the menu CosmoSPHERES MS General to find the function block “my mass”


global variables
type: float name: mymass initial value: 0
init


loop
mymass = my mass

MS2014 Zone 1- Check Mass, continued

- Add a debug statement
  - You can use debug statements to print variables as follows. Use the following symbols for each data type:

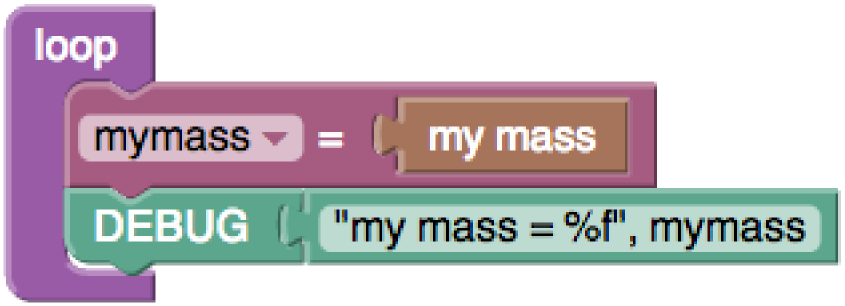
Data type	symbol
Float	%f
Int	%d

  - Use the following format inside the debug block:

"text text text symbol", variable

For this example:

"my mass =%f**",** mymass

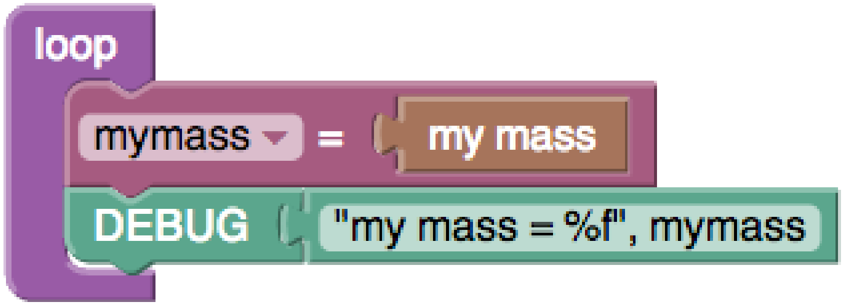


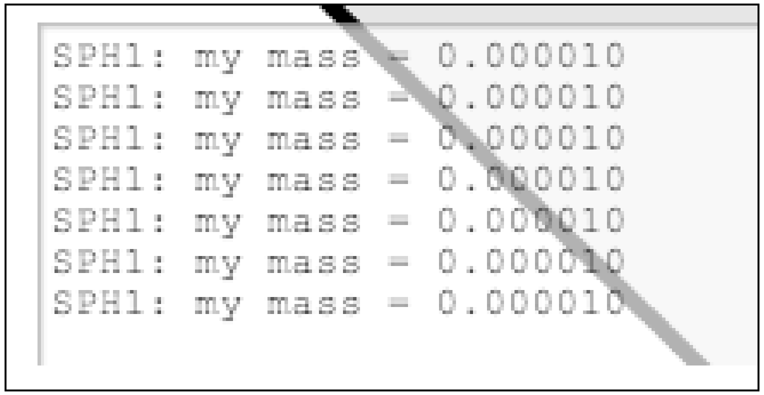
MS2014 Zone 1- Check Mass, continued

- Simulate and view results

- This example code will return the initial SPHERES mass (since the SPHERE has not collected any debris)
- In the console, you should see the message:

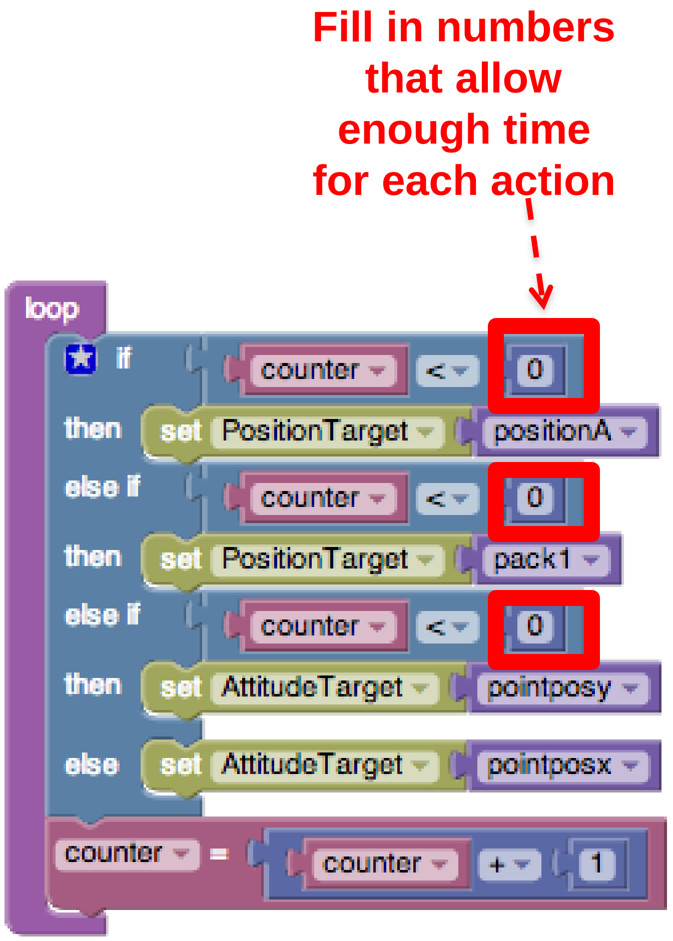
  SPH1: my mass= 0.00001


```
loop
    mymass = my mass
    DEBUG ( "my mass = %f", mymass
```


```
SPH1: my mass = 0.000010
SPH1: my mass = 0.000010
SPH1: my mass = 0.000010
SPH1: my mass = 0.000010
SPH1: my mass = 0.000010
SPH1: my mass = 0.000010
SPH1: my mass = 0.000010
```

Zone 2 – Pick up Laser Pack- example (using time counters)

- To pick up a Laser pack
  - Stop the SPHERES
    - within 0.05 m of the location of the pack
  - Rotate SPHERES 90˚
    - to pick up the laser pack
- Try completing the example shown:
  - Goal: Send the SPHERE through the debris field without any collisions and then pick up Pack 1
  - Declare arrays and variable for:
    - PositionA, pack1, pointposy, pointposx, counter
    - Pick coordinates for each array based on the game layout
  - Fill in the numbers in the counters to allow enough time for the SPHERE to complete each action

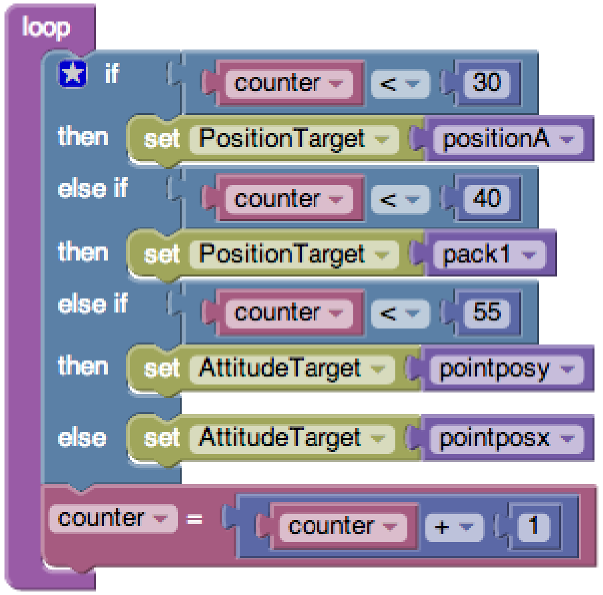


```
loop
  if counter < 0
  then set PositionTarget positionA
  else if counter < 0
  then set PositionTarget pack1
  else if counter < 0
  then set AttitudeTarget pointposy
  else set AttitudeTarget pointposx
  counter = counter + 1
```

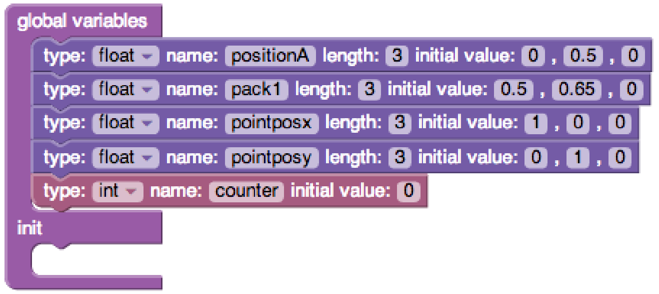
Zone 2 – Pick up Laser, continued

- Simulate and view your program!
- All criteria must be met for the laser pack to be picked up.
  - The SPHERE should:
    - Go to laser pack 1 and stop
    - Rotate 90 degrees to pick up the laser pack.
      - The SPHERES must have enough time to rotate 90 degrees or the pack will not be picked up
      - The 90 degree rotation must happen while the SPHERE is stopped at the laser pack location
  - The laser pack will disappear to indicate that it has been picked up
- If needed go back and edit your program and try again.

Possible solution



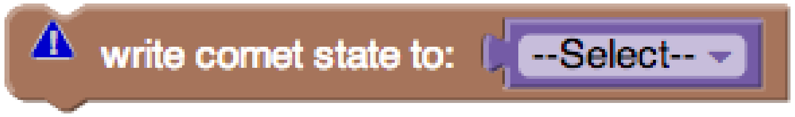
```
loop
  if counter < 30
  then set PositionTarget positionA
  else if counter < 40
  then set PositionTarget pack1
  else if counter < 55
  then set AttitudeTarget pointposy
  else set AttitudeTarget pointposx
  counter = counter + 1
```

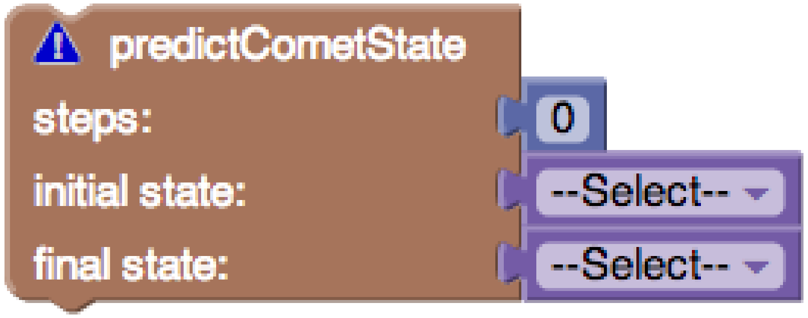


```
global variables
  type: float name: positionA length: 3 initial value: 0 , 0.5 , 0
  type: float name: pack1 length: 3 initial value: 0.5 , 0.65 , 0
  type: float name: pointposx length: 3 initial value: 1 , 0 , 0
  type: float name: pointposy length: 3 initial value: 0 , 1 , 0
  type: int name: counter initial value: 0
init
```

Zone 3 – Comet Functions

- The functions *getCometState* and *PredictCometLocation* provide information useful for
  - Following the comet
  - Shooting at the comet





- *getCometState[6]* retrieves the following information about the Comet and is provided in an array of size 6:

Position	(x,y,z)	[0] [1] [2]
Velocity	(vx,vy,vz)	[3] [4] [5]

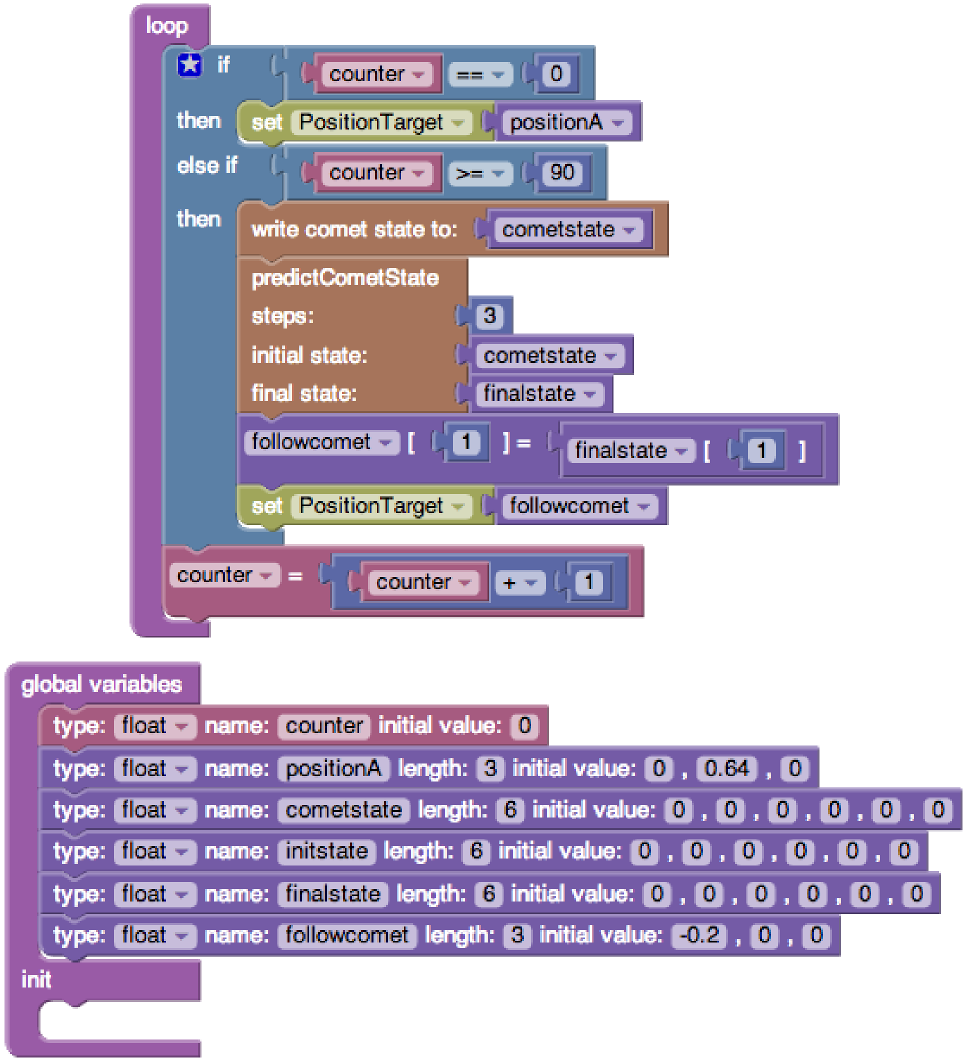
- *predictCometState[6]* stores the predicted location of the comet in a selected number of Steps **(**seconds) into **finalState** based on linear extrapolation using the current velocity
  - The final state information can be used to define Satellite coordinates

Zone 3 – Predict Comet Location- example

- In this example getCometState is to used to define initial state in the predictCometState function
- The y coordinate from the predicted “final state” is assigned to the array “followcomet”

`followcomet[1]=finalstate[1]`

- The y coordinate for “followcomet” is updated every second after counter>=90
- The x and z coordinates of the array “followcomet” are constant and are assigned on the init page.

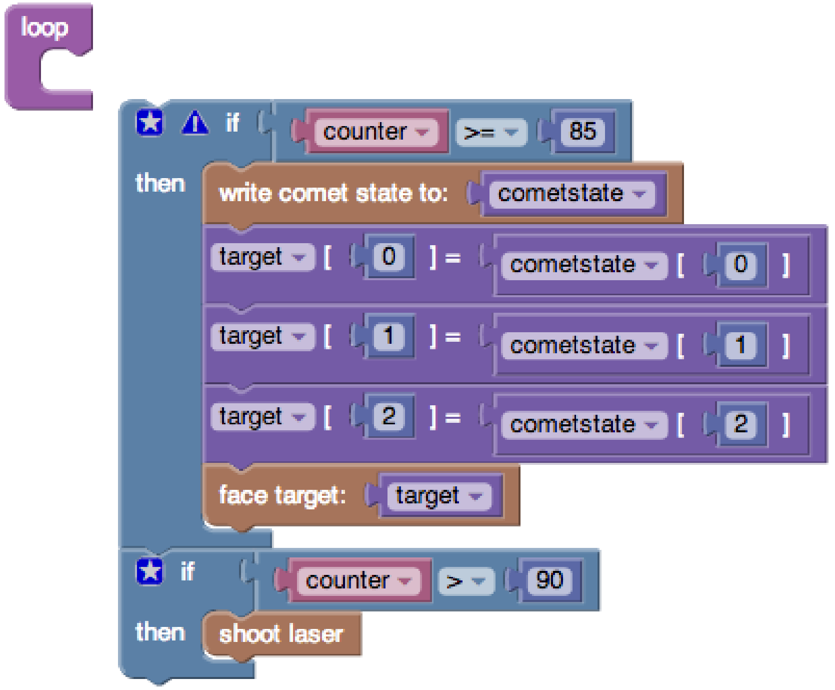


```
loop
  if counter == 0
  then set PositionTarget positionA
  else if counter >= 90
  then write comet state to: cometstate
       predictCometState
         steps: 3
         initial state: cometstate
         final state: finalstate
       followcomet [ 1 ] = finalstate [ 1 ]
       set PositionTarget followcomet
  counter = counter + 1

global variables
  type: float name: counter initial value: 0
  type: float name: positionA length: 3 initial value: 0 , 0.64 , 0
  type: float name: cometstate length: 6 initial value: 0 , 0 , 0 , 0 , 0 , 0
  type: float name: initstate length: 6 initial value: 0 , 0 , 0 , 0 , 0 , 0
  type: float name: finalstate length: 6 initial value: 0 , 0 , 0 , 0 , 0 , 0
  type: float name: followcomet length: 3 initial value: -0.2 , 0 , 0
init
```

Zone 3 – Shoot Laser

- In this example:
  - getCometState is to used to assign values to the array “target”
  - the “facetarget function is used to turn the SPHERE toward the target
  - Note that the SPHERE needs to be given time to face the target before firing
  - The “shoot laser” function is used to fire the laser

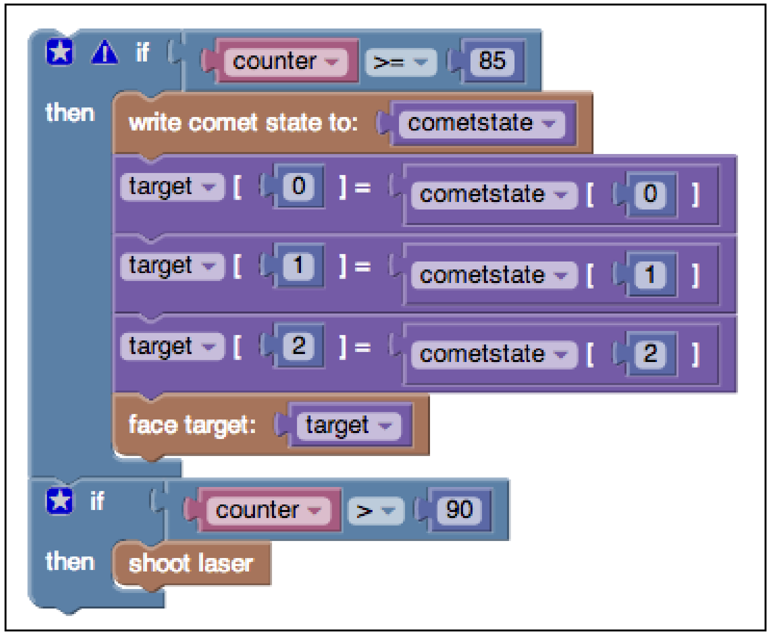


```
loop
  if counter >= 85
  then write comet state to: cometstate
       target [ 0 ] = cometstate [ 0 ]
       target [ 1 ] = cometstate [ 1 ]
       target [ 2 ] = cometstate [ 2 ]
       face target: target
  if counter > 90
  then shoot laser
```

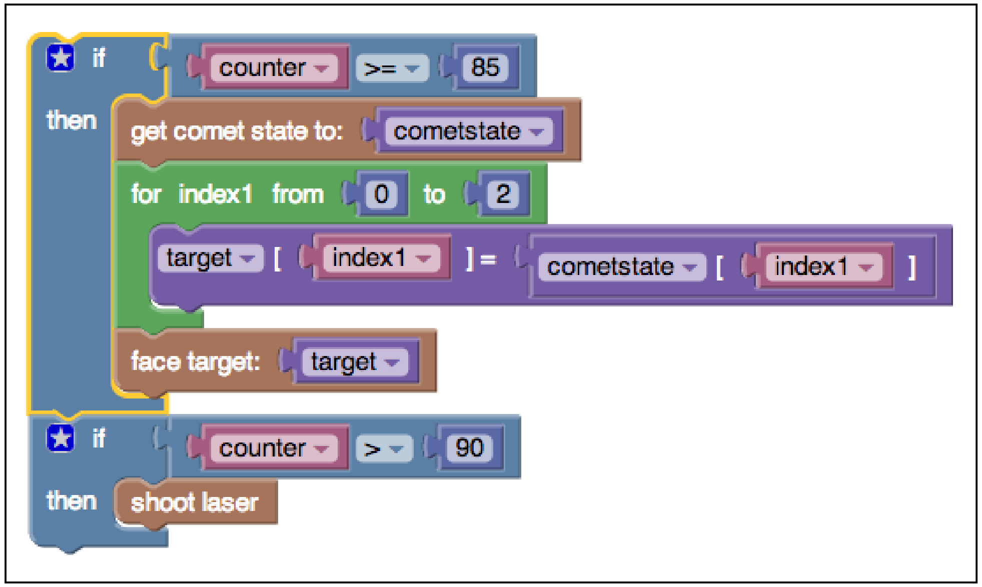
- IMPORTANT: This example only shows that shoot laser portion of the code. Code needs to be added to command the SPHERE to pick up the laser pack in order to make it possible for the SPHERE to fire laser shots

Zone 3 – Shoot Laser, alternate code

- Two versions of the shoot laser portion of code are shown here:
- Both will give the same result
- The example on the right uses a **loop function** to assign values to the array “target”



```
if counter >= 85
then
    write comet state to: cometstate
    target [ 0 ] = cometstate [ 0 ]
    target [ 1 ] = cometstate [ 1 ]
    target [ 2 ] = cometstate [ 2 ]
    face target: target
if counter > 90
then
    shoot laser
```



```
if counter >= 85
then
    get comet state to: cometstate
    for index1 from 0 to 2
        target [ index1 ] = cometstate [ index1 ]
    face target: target
if counter > 90
then
    shoot laser
```

Review

- Congratulations!
  - You know the basics about the game
  - You have some experience with how to use the Game Specific Functions!