



Zero Robotics ISS Programming Challenge Middle School Summer Program 2015:



Conducting Optical Research on Nearby Asteroids (CORONA)

GAME MANUAL

V1.2

2015-June-06



To: Zero Robotics Teams
Re: CORONA program

Attention to all teams:

NASA has observed a recent spike in the number of asteroids knocked loose from the asteroid belt and into close proximity to Earth. Using highly sophisticated algorithms, they have used the mass of these space rocks to predict the composition of these asteroids with great accuracy. However, there exists amongst these rocks a small percentage that are of comparable mass percentages but do not fit the model. Eager geologists believe scientists may have discovered a new element and immediately rally for further study.

NASA calls upon its fellow scientists at MIT for a plan of action. With the help of these researchers, the joint CORONA program was born. CORONA (Conducting Optical Research On Nearby Asteroids) requires the use of MIT's most recent project, SPHERES, to capture visuals of the closest asteroids to Earth's atmosphere.

MIT researchers are fairly confident that they have stumbled upon something novel in these asteroids. However, some believe that there may actually be ice on these rocks and not a new element at all. The contrasting opinions lead to the creation of two separate SPHERES teams. Both teams will be using their respective satellites to take up-close photos of the asteroids at specific points of interest, as determined by NASA.

Soon after launch, however, NASA catches sight of incoming solar flares at the exact location where the SPHERES are intending to intercept the asteroids. The satellites risk imminent mechanical issues and loss of their stored photos if they impact with the solar flare while turned on. The MIT teams have to decide whether they want to use the shadow zone of the asteroid for protection or temporarily power down when a solar flare is near.

The choice is a tough one, but one that must be made. Each team is counting on its satellite to find visual proof of why either water or a new element exists on these space rocks. Proof of either would be invaluable, but conclusive evidence is essential.

As a SPHERES expert, your skills will be in high demand. GOOD LUCK!



Alvar Saenz-Otero
MIT SPHERES Lead Scientist



Contents

1	GAME OVERVIEW	4
1.1	GAME LAYOUT	4
1.2	SATELLITE	5
1.2.1	ZR User API	5
1.2.2	Time	5
1.2.3	Fuel	5
1.2.4	Code Size	6
1.3	INITIAL POSITION	6
1.3.1	PlayerID	7
1.4	GAME PLAY	7
1.4.1	Picture Taking	8
1.4.2	Memory Upgrade Packs	10
1.4.3	Solar Flares	10
1.4.4	Upload	11
1.4.5	Collision with Asteroid	12
1.4.6	End of Game	12
1.5	ITEM COLLECTION	12
1.6	OUT OF BOUNDS	14
1.7	NOISE	14
2	SCORING	14
3	TOURNAMENT	15
3.1	REGIONAL SIMULATION COMPETITION	15
3.1.1	Competition Periods	15
3.1.2	Submitting Code	15
3.1.3	Competition Format – Regional Competition	15
3.2	COLLABORATION FOR ISS FINALS	16
3.3	ISS FINAL COMPETITION	16
3.3.1	Overview and Objectives	16
3.3.2	Competition Format	16
3.3.3	Scoring Matches	17
4	SEASON RULES	17
4.1	TOURNAMENT RULES	17
4.2	ETHICS CODE	18
5	ZR USER API	18
6	LISTS OF FIGURES AND TABLES	21
6.1	LIST OF FIGURES	21
6.2	LIST OF TABLES	21
7	REVISION HISTORY	22



1 Game Overview

Matches will be played between two SPHERES satellites, controlled by code written by two different teams. Satellites start the game facing away from the asteroid. Each team will attempt to take pictures of special points of interest (POI). These points of interest reset once every minute. The SPHERES can hold up to two pictures in its memory at a time. During the game, teams have the option of picking up a memory upgrade pack, which allows players to store one extra picture per memory upgrade pack, before needing to upload. There are two zones around the asteroid which determine the number of points each picture is worth. Points are received only after uploading the pictures stored in each satellite's memory. Throughout the game, there are solar flares that can corrupt the pictures in memory, and if precautions are not taken, can also damage the satellite. Satellites are completely safe only in the shadow zone.

1.1 Game Layout

The Zero Robotics Middle School Summer Program begins with competitions in simulation. The competition mimics the operational volume available aboard the International Space Station, where the ISS finals will be conducted in August 2015. The middle school game takes place in 2D. The game arena is a plane with only X and Y Cartesian dimensions. It does not matter what Z coordinate you command the satellite to move to. Because instructions to leave the plane are ignored, you can command $Z = 0$ or any other convenient value and the controller will set the correct value automatically. The game arena encompasses the complete area where the SPHERES satellites can operate as shown in Figure 1. However, the game is played in a smaller area called the Interaction Zone. If a satellite leaves the Interaction Zone, it may still be within the arena operational area, but it will be considered out of bounds.

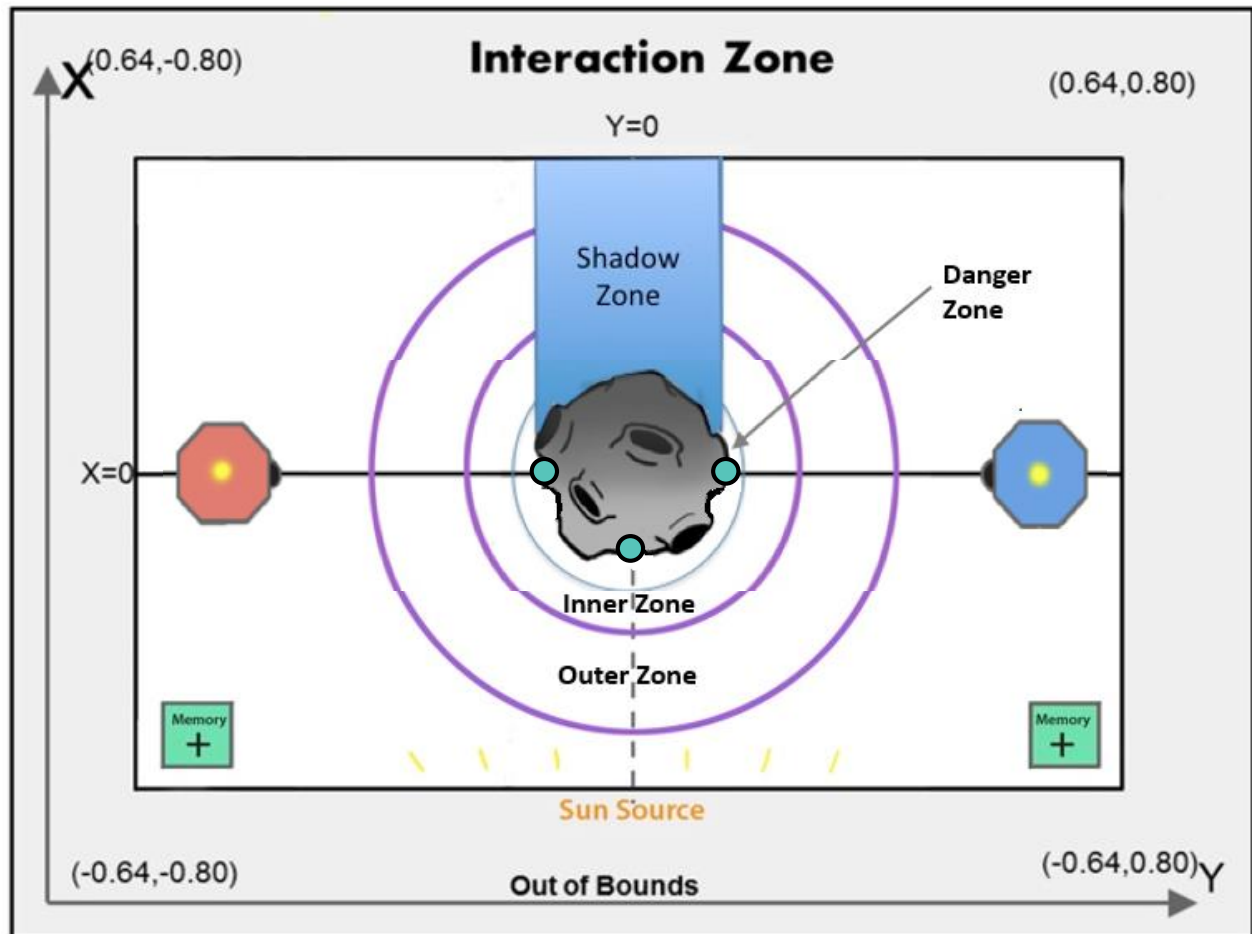


Diagram not to scale

Figure 1 Game Overview



The dimensions of the *Interaction Zone* are:

Table 1 Interaction Zone Dimensions

X [m]	[-0.64 : +0.64]
Y [m]	[-0.80 : +0.80]
Z[m]	0.0

Within the interaction zone there are three zones: Danger, Inner, and Outer zones.

Satellites should stay out of the Danger Zone to avoid collisions with the asteroid. See section 1.4.5 for details.

Inner and Outer zones affect the number of points a picture is worth.

The Shadow zone represents the area safe from Solar Flares.

Table 2 Zone Radii Positions

Danger min radius (m)	0.2
Danger max radius or Inner min radius (m)	0.31
Inner max radius or Outer min radius (m)	0.42
Outer max radius (m)	0.53

Table 3 Shadow Zone Dimensions

X [m]	[0.0 : +0.64]
Y [m]	[-0.2 : +0.2]
Z[m]	0.0

1.2 Satellite

Each team will write the software to command a SPHERES satellite to move in order to complete the game tasks. A SPHERES satellite can move in all directions using its twelve thrusters. (For the middle school game, the ability to move to a different Z-coordinate has been disabled.) The actual SPHERES satellite, like any other spacecraft, has a fuel source (in this case, gaseous carbon dioxide) and a power source (in this case, AA battery packs). These resources are limited and must be used wisely. Therefore, the players of Zero Robotics are limited in the use of fuel and batteries by virtual limits within the game. This section describes the limits to which players must adhere to in order to wisely use virtual SPHERES resources.

1.2.1 ZR User API

Game specific functions, along with the standard ZR User API functions, are provided in Section 5 of this manual. The various functions used to control the SPHERES satellite in ZeroRobotics are located in a document titled “ZR User API” on the ZeroRobotics website (zerorobotics.mit.edu) under “Tournaments” > “Zero Robotics Middle School Summer Program 2015” > “Game Documents” > “Game Functions”.

1.2.2 Time

Players have 180s to take and upload as many photos as possible. After 180s scores will be final and compared.

1.2.3 Fuel

Each player is assigned a virtual fuel allocation (Table 4) which is the total sum of fuel used in seconds of individual thruster firing. Calling the function `Get remaining fuel` (`float getFuelRemaining()`) returns remaining fuel. Once the allocation is consumed, the satellite will not be able to respond to SPHERES control commands. It will fire thrusters only to avoid leaving the Interaction Zone or colliding with the other satellite.



Table 4 Fuel Allocation

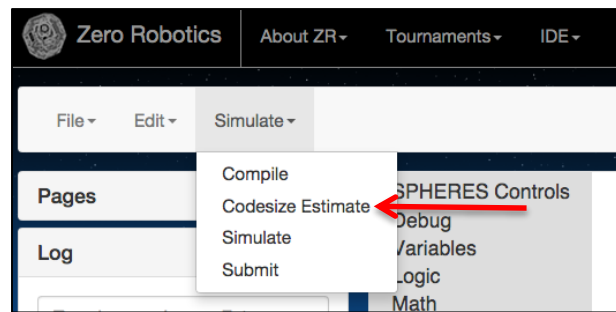
Fuel Allocation [s]	60s
---------------------	-----

The virtual fuel allocation is consumed any time the thrusters are fired. Potential reasons include:

- Motion initiated by player
- Motion initiated by the SPHERES controller to avoid leaving the Interaction Zone (see section 1.6)
- Motion initiated by the SPHERES controller to avoid a collision with the other satellite

1.2.4 Code Size

A SPHERES satellite can fit a limited amount of code in its memory. Each project has a specific code size allocation. When you compile your project with a code size estimate, the compiler will provide the percentage of the code size allocation that your project is using. Formal competition submissions require that your code size be 100% or less of the total allocation. To check your project's code size –open your project in the IDE then select “Code Size Estimate” under the simulation tab as shown in the figure below. The percent usage will be displayed in the Log.

**Figure 2: How to Check Project Code Size**

1.3 Initial Position

Each satellite starts on the Y axis on opposite sides of the asteroid.

Teams should write code assuming that their player is the blue SPHERES. It is not necessary for a teams code to account for the possibility of being either red SPHERES or blue SPHERES. This adjustment will be made automatically.

The SPHERES satellites are deployed at:

Table 5 SPHERES Satellite Deployment Locations

Blue	
X [m]	0.0
Y [m]	0.6
Z[m]	0.0
Red	
X [m]	0.0
Y [m]	-0.6
Z[m]	0.0

Table 6 Asteroid and SPHERES Measurements

Asteroid radius (m)	0.2
SPHERE radius (m)	0.11

1.3.1 PlayerID

Users will identify themselves as “playerID = 0” and opponents as “playerID = 1” for all games, whether or not they are the red SPHERES satellite or the blue one.

1.4 Game play

A set of 3 points of interest (POIs) will be visible on the surface of the asteroid (sunny side) at all times during the game. The POIs are "reset" as indicated by a change in color *every 60 seconds starting at time =0s*. You can only take one picture of each POI from each zone until the POIs are reset. See section 1.4.1.

The visible POIs are assigned an identification number (ID) based on their location. The POI ID #s are 0, 1 or 2. The POI ID # is used in the function for taking pictures. See section 1.4.1

The location of the POIs are shown in Figure 3 and Table 7. Calling the game function `getPOILoc(float pos[3], int id)` will also return the location of each visible POI.

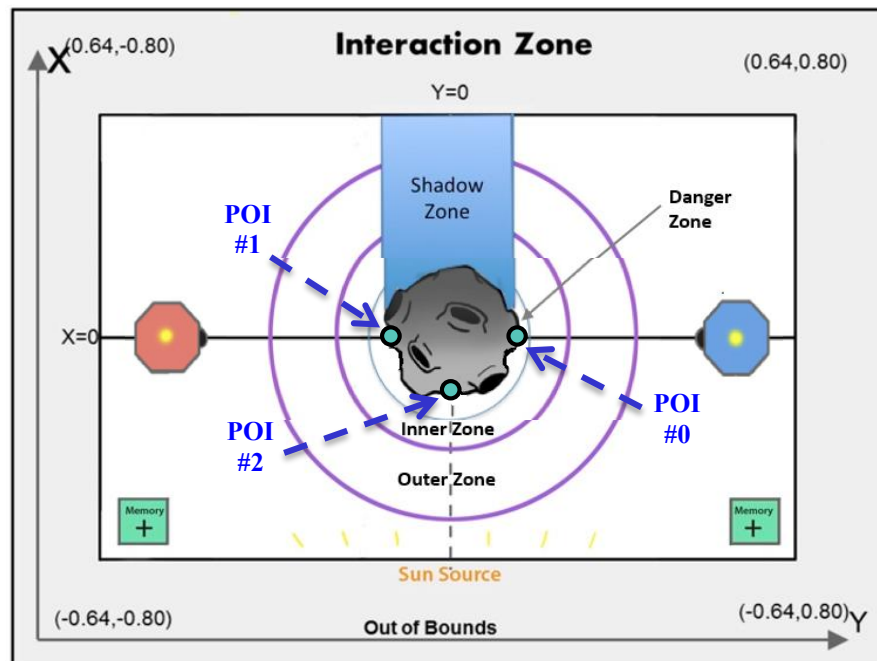
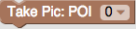
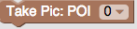


Figure 3 POI Locations

Table 7 POI Locations

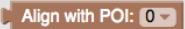
POI #	(x,y,z)
0	(0.0, 0.2, 0.0)
1	(0.0, -0.2, 0.0)
2	(-0.2, 0.0, 0.0)

1.4.1 Picture Taking

Pictures are taken by calling the function  (void takePic(int poiID)). The camera will be disabled for 3 seconds each time after the  (void takePic(int poiID)) function is called.

In order for a picture to be valid and earn full points when uploaded all the following qualifications must be met:

1. The satellite camera must be facing the POI with a tolerance of +/- 46 degrees. (Tolerance shown in Figure 4)

Game function  (bool alignLine(int poiID)) Returns true if the SPHERE is facing the POI whose ID is given within the proper tolerances.

2. The satellite must be in the Inner or Outer Zone

There are two possible orbits from which the satellite can take pictures: a low orbit in the Inner Zone and a high orbit in the Outer Zone (dimensions listed above in Table 2). For the satellite to be in the correct orbit, the center point of the satellite must be within the bounds of the respective Zone (Figure 1). Scoring will be determined by which orbit the satellite is in; more points will be awarded for pictures taken in the high orbit. The point values awarded for pictures taken in the two orbits (inner zone and outer zone) are provided in Table 8.

Table 8 Picture Values in Inner and Outer Zones

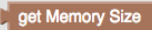
	Points
Inner Zone	2
Outer Zone	3


3. The satellite must be properly aligned with the POI (Table 9 and Figure 4). To capture an acceptable image of the POI the SPHERE must be properly aligned with the POI within acceptable tolerances. This tolerance is different for the Inner Zone and the Outer Zone. Table 9 and Figure 4 describe for each picture taking zone the acceptable tolerances in degrees from a picture taking position located directly opposite the POI. More formally, the angle between the line connecting the center of the sphere and the POI and the line connecting the POI and the center of the asteroid must be smaller than the Max Angle for the zone the satellite is in as described in Table 9.

Table 9 Max Angle allowed (in degrees)

Inner Zone	+/-46
Outer Zone	+/-23

4. Pictures of a single POI cannot be taken from the same orbit within each 60s window.
There are six possible pictures to take in each 60s window, a picture of each of the POIs can be taken from each orbit once. These limitations are reset after each 60s window.
5. Camera must have a memory slot available to store the picture
The satellite will start with the ability to store 2 photos (2 memory slots) at a time unless it has obtained an upgrade from a memory pack (see 1.4.2). The camera will be disabled once the photo storage is full until the pictures are uploaded.

The function  (int getMemorySize()) returns the current limit of picture storage.

The function  (int getMemoryFilled()) returns the number of valid pictures (taken from the right distance and angle from the poi) currently saved in camera.

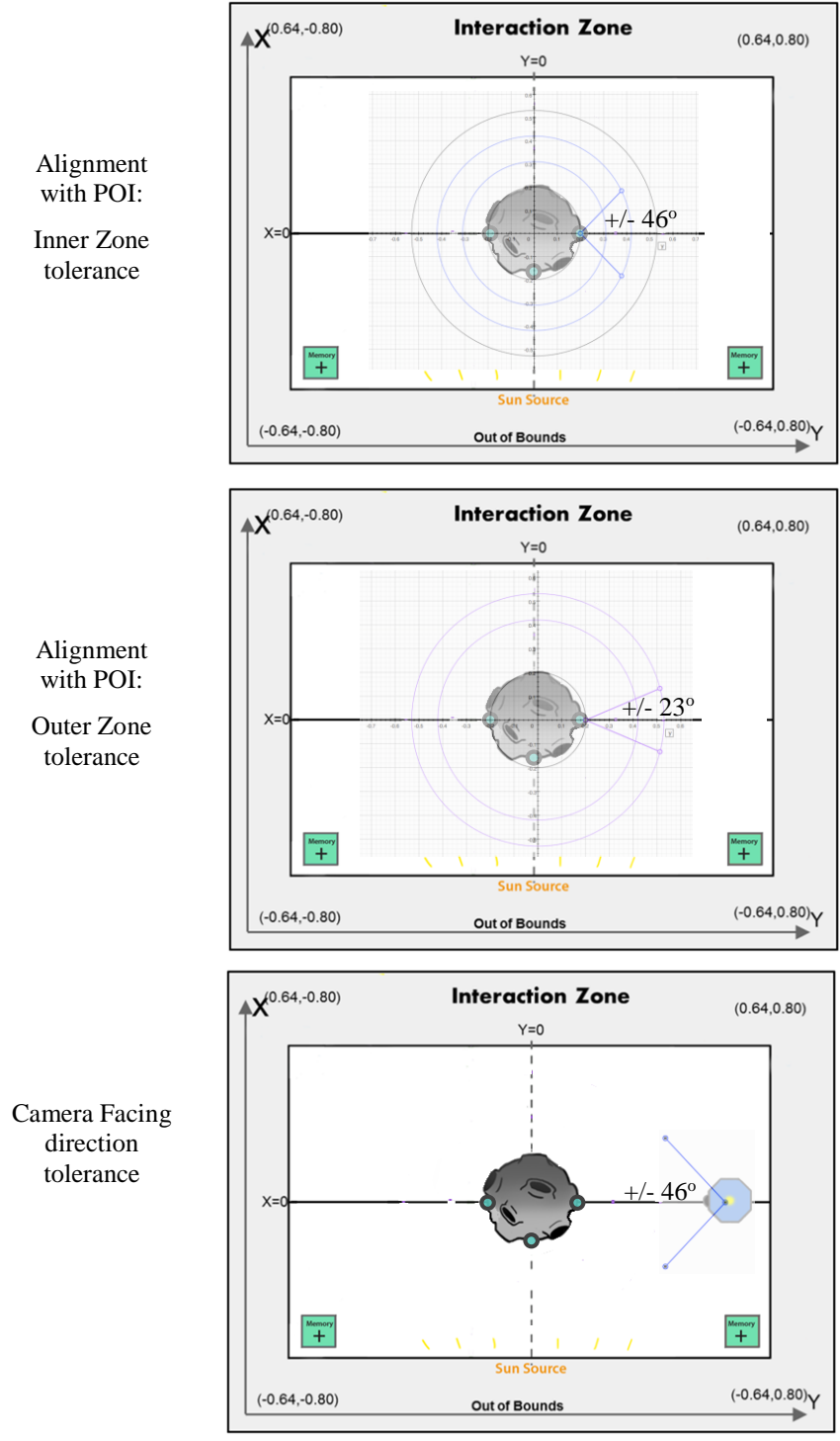


Figure 4: Picture Taking Tolerances (diagrams not to scale)

1.4.2 Memory Upgrade Packs

Memory Upgrade packs allow players to store more pictures on their satellite. Memory Upgrade packs are located on the corners of the sunny side of the interaction zone. Memory packs will remain in play throughout game play. Once a memory pack has been taken by a satellite, it is then in the possession of that satellite and may not be picked up by the other satellite. A satellite can collect up to two memory packs. Each pack gives the satellite enough memory to hold one more photo. Calling the function `player: (Me) has pack: 0` (`bool hasMemoryPack(int playerId, int packID)`) returns “true” if specified player has specified memory pack. Memory Upgrade Pack Locations are shown in Table 10.

Table 10 Memory Upgrade Pack Locations

	2D
Memory Pack 0 (ID = 0)	
X [m]	-0.5
Y [m]	-0.6
Z[m]	0.0
Memory Pack 1 (ID = 1)	
X [m]	-0.5
Y [m]	0.6
Z[m]	0.0

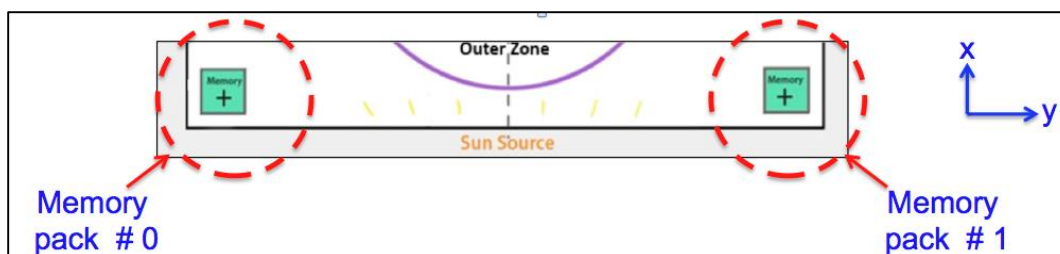


Figure 5: Memory Pack Locations

To collect Memory Upgrade packs see section 1.5 ITEM COLLECTION.

1.4.3 Solar Flares

Solar Flares pose a danger to satellites. Solar Flares occur randomly 30s after the start of the game. (Note flares are separated by at least 30 seconds from each other.) Each Solar Flare lasts three seconds.

Table 11 Number of Solar Flares

Number of Solar Flares	2
------------------------	---


All satellites exposed to the solar radiation for any period of time lose the photos stored in their memory.

Satellites exposed to the solar radiation will lose one point every second they are exposed to the solar radiation, unless the satellites are powered off prior to exposure with the solar radiation.

While the satellite is powered off, players:

- Reduce Damage (lose half the points of satellites powered on)
- Lose any pictures currently stored on satellite
- Must wait five seconds for their instruments to turn on and warm up
- Cannot take pictures
- Cannot stop their satellites from drifting (collisions may occur)


To power off:

- Game function  (void turnOff()) must be called

To power back on:

- Game function  (void turnOn()) must be called

The other way to prevent damage is to seek shelter in the shadow zone. Players are protected from solar radiation in the shadow zone, preserving pictures and points. To be deemed safe, the center point of the satellite must be in the shadow zone.

Players will have a 30 second warning before a solar flare arrives. This warning can be received by calling the function:  (int getNextFlare()). During the 30 second period prior to the next solar flare int getNextFlare() returns the number of seconds until the next solar flare. Calling this function any time outside this 30 second window will return -1. Calling this function during the solar flare will also return -1.

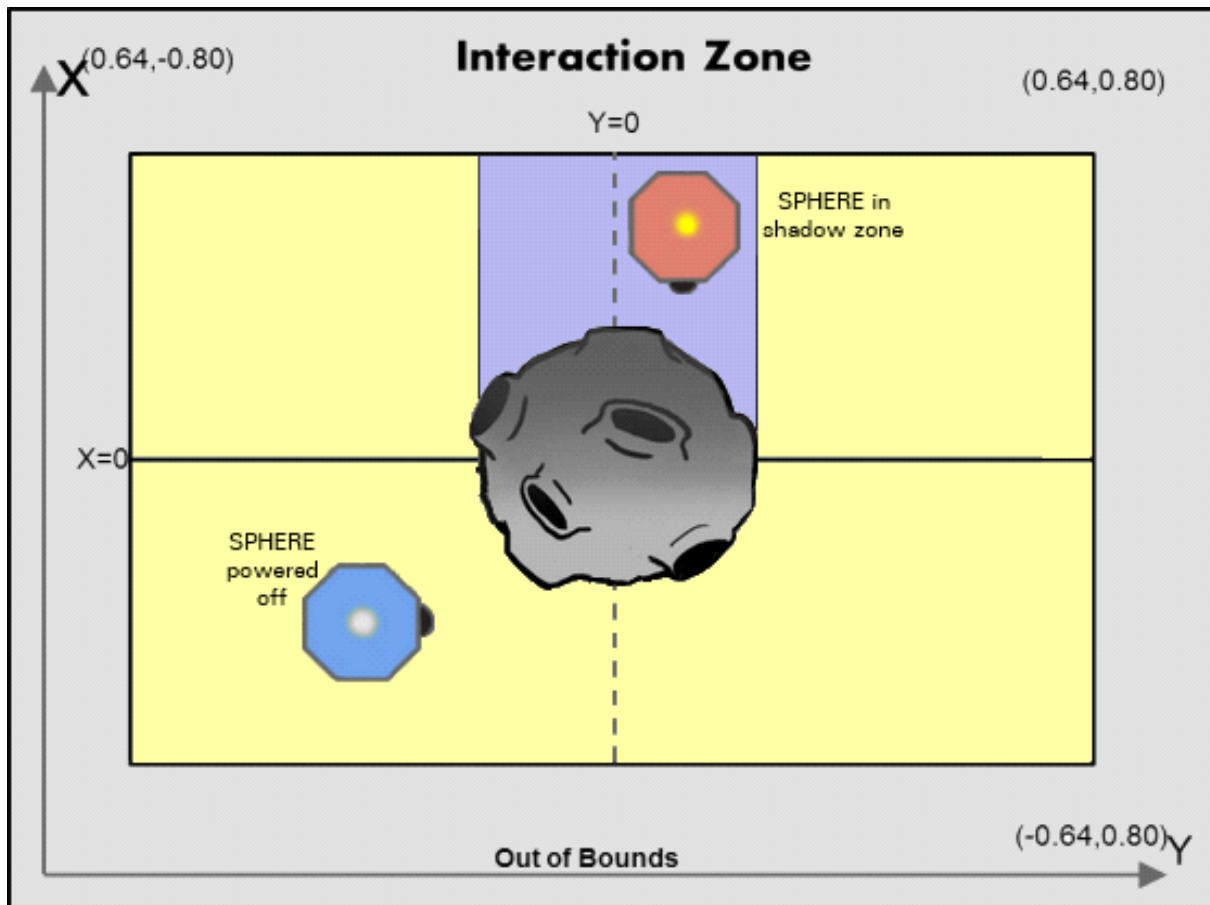


Diagram not to scale

Figure 6 Solar Flare

1.4.4 Upload

Once the satellite's memory is full, the pictures must be uploaded. Pictures must be uploaded to score full points for taking the picture. Without uploading, only 0.1 points will be awarded for successfully taking a photo and .01 points will be awarded for attempting to take a photo.

During the upload process the camera will be disabled for three seconds.

There is a Discover Bonus for the first satellite to upload a picture of a specific point of interest within a cycle. The Discover Bonus is +0.5. The Discover Bonus resets with each 60s POI rotation cycle.

To upload photos:

- The center of the SPHERE must be either outside of the Danger Zone and both Picture Taking zones, or in the Shadow Zone
- The game function `uploadPic()` (void uploadPic()) must be called

1.4.5 Collision with Asteroid

While it is not possible to collide with the other satellite, collision with the asteroid is possible. If the center point of your satellite enters the Danger Zone, it is considered a collision with the asteroid.

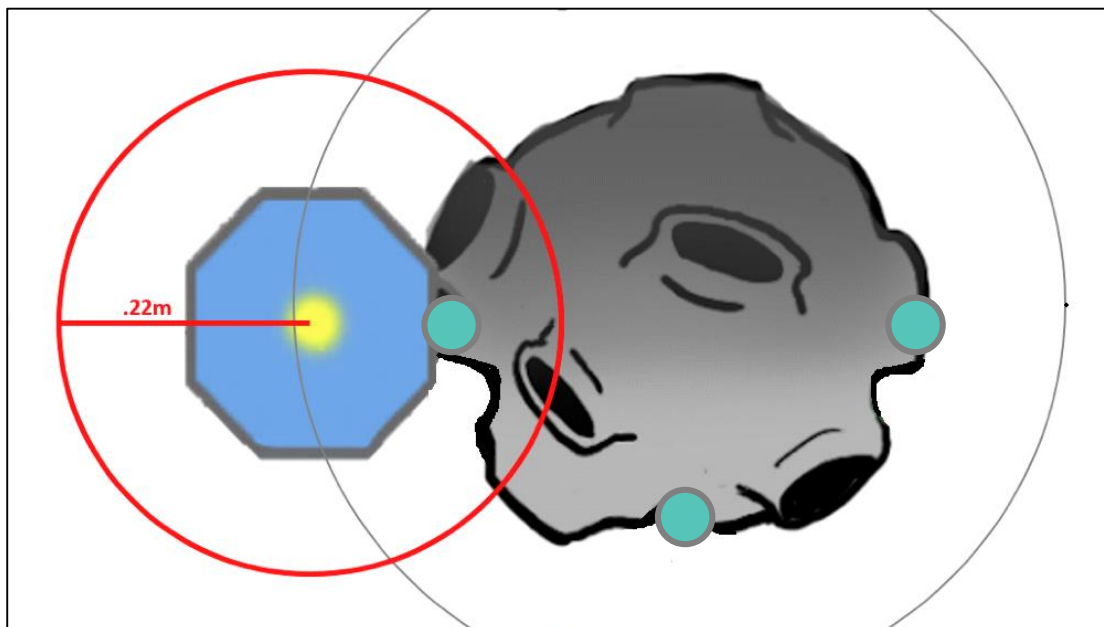


Diagram not to scale

Figure 7 Asteroid Collision

If the satellite crashes into the asteroid:

- The fuel percentage will be subtracted by 4% for each second in the asteroid.
- Points will be lost equal to the number of POI affected in crash site multiplied by 0.1.

The crash site is all points on the asteroid within a satellite diameter (.22m) of the satellite's center point.

In the figure above, the portion of red circle that intersects with the asteroid represents the crash site. All points of interest within this crash site will be counted against the team whose satellite crashes into the asteroid.

1.4.6 End of Game

The game ends when time runs out.

1.5 Item Collection

To increase the memory capacity of the satellites, teams have the opportunity to collect memory upgrade packs found in the 2 corners of the interaction zone closest to the sun.

Table 12 Memory Upgrade Pack Locations (repeated)

Memory Pack 0	
X [m]	-0.5
Y [m]	-0.6
Z[m]	0.0
Memory Pack 1	
X [m]	-0.5
Y [m]	0.6
Z[m]	0.0

In order to pick up an item, you need to perform a spinning maneuver (see Figure 7). The steps to collect the Memory Upgrade packs are:

- Position the satellite within 0.05m of the item's center.
 - The satellite's velocity must be less than 0.01m/s.
 - The satellite's angular velocity must start at less than $\pm 2.3^\circ/\text{s}$.
- Rotate the satellite more than $\pm 90^\circ$ along about the Z axis. Do not attempt to rotate faster than $\pm 80^\circ/\text{s}$.

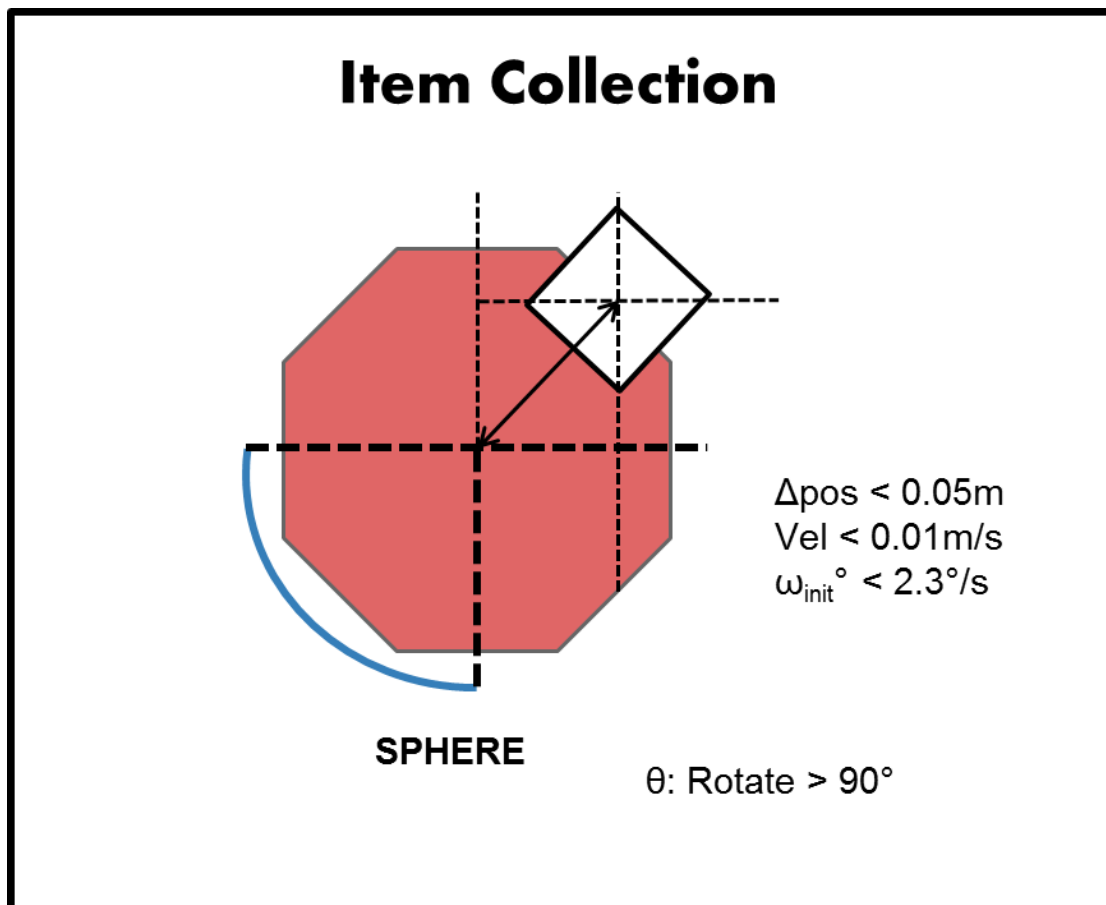


Figure 8 Maneuver to Collect Item

1.6 Out of Bounds

You must remain within the boundaries of the Interaction Zone to avoid a fuel penalty.

If you exit the interaction zone, the fuel percentage will be subtracted by 4% for each second spent out of bounds.

1.7 Noise

It is important to note that although the two competitors in a match will always be performing the same challenge and have identical satellites, the two satellites may be affected by random environmental variations in different ways, resulting in small or even large differences in score. This is fully intended as part of the challenge and reflects uncertainties in the satellite behavior models. The best performing solutions will be those that can overcome these variations and still perform the task at hand.

2 Scoring

Your satellite begins with a score of 6 points. Your final score is based on the pictures you take and upload, minus the damage obtained from solar flares and collisions. A picture taken from an outer orbit is worth more than a picture taken from an inner orbit. Your team will also receive a bonus for being the first satellite to photograph a point of interest from each orbit. The more pictures you take and upload, the higher your final score will be; however, points will be deducted from your score for every second your satellite is exposed to solar radiation.

The scoring calculation is as follows:

Table 13 Point Values

Number of points at time = 0	6
Pictures attempted (both valid and invalid pictures)	.01
Non-uploaded valid photos	0.1
Uploaded: Taken in Inner Zone	2
Uploaded: Taken in Outer Zone	3
Collision with Asteroid:	-0.1
Points deducted per second per POI contacted	
Discover Bonus	+0.5
Solar Flare:	-1
Points deducted per second: SPHERE powered, outside shadow zone	
Solar Flare:	-0.5
Points deducted per second: SPHERE un-powered, outside shadow zone	
Solar Flare:	0
Points deducted per second: SPHERE inside shadow zone	
To avoid ties: At the end of the match, to avoid ties, points will be subtracted from the score based on the following calculation:	
(satellite distance from the center of asteroid) x (0.00001)	

3 Tournament

A Zero Robotics tournament consists of several phases called *competitions*. The following table lists the key deadlines for the 2015 tournament season:

Table 14 Tournament Key Dates

Date (2015)	Schedule 1	Schedule 2
June 8 (Mon)	Program begins	
June 15 (Mon)		
June 26 (Fri), 5 PM	Code submittal deadline: <i>Practice Regional Competition</i>	
July 3 (Fri), 5 PM	Code submittal deadline: <i>Regional Competition</i>	
July 6 (Mon)		Program begins
July 9 (Thurs), 5 PM	Code submittal deadline: <i>ISS Competition</i>	
July 13 (Mon)		
July 24 (Fri), 5 PM		Code submittal deadline: <i>Practice Regional Competition</i>
July 31 (Fri), 5 PM		Code submittal deadline: <i>Regional Competition</i>
Aug 6 th (Thurs), 5PM		Code submittal deadline: <i>ISS Competition</i>
Mid- August (TBD)	ISS Finals Event	ISS Finals Event

3.1 Regional Simulation Competition

3.1.1 Competition Periods

The program starts with two phases of regional simulation competition:

- **Practice Regional Competition** At the end of Week 3 of the summer program, teams will submit their code and a competition will be run. The results of this competition are not official and are intended to guide teams in improving their code during Week 4. The submission deadline is 5 PM local time on the Friday of Week 3 (the date may vary by region.)
- **Regional Competition** At the end of Week 4 of the summer program, teams will submit their updated code and a competition will be run. The results of this competition determine the regional 1st, 2nd, and 3rd place champions. The submission deadline is 5 PM local time on the Friday of Week 4.

3.1.2 Submitting Code

To enter a program in a competition the team must use the Submit tool located under the Simulate menu on the IDE page of the ZeroRobotics website. You may change your submission as many times as you like before the submission deadline, but only the most recently submitted program before the deadline will be used. No programs submitted after the deadline will be accepted unless the Zero Robotics staff determines that emergency circumstances made timely submission impossible.

3.1.3 Competition Format – Regional Competition

The regional competition will be a round robin, with every team playing every other at least once. Each team will play approximately half its matches with each satellite (blue and red). The team with the most wins will be the champion. In the event of a tie, the team that won the most head-to-head matches against the other tied team(s) will be the champion. If this procedure fails to resolve a tie, the tied team with the highest total score (that is, the scores from all of its matches added together) will be the champion. The results of regional competitions will be released by 8 AM ET



on the Monday after the competition submission deadline. The Zero Robotics team will release them earlier if possible. All regional results may not be released simultaneously.

3.2 Collaboration for ISS Finals

During the first several days of week 5 of the summer program all teams in each region will have an opportunity to collaborate to try to improve their 1st place regional winner's code prior to ISS submittal deadline. Teams from the same region are encouraged to try to beat the regional winner's code and then share their ideas with the regional winner. The regional winner will submit the final code from their region for the ISS Competition. The submission deadline is 5pm on the Thursday of Week 5.

3.3 ISS Final Competition

The final code submitted by the regional winner from each region will compete in the ISS finals. The finals will take place aboard the International Space Station with live video transmission. All teams will be invited to watch the live broadcast.

3.3.1 Overview and Objectives

Running a live competition with robots in space presents a number of real-world challenges that factor into the rules of the competition. Among many items, the satellites use battery packs and CO₂ tanks that can be exhausted in the middle of a match, and the competition must fit in the allocated time. This section establishes several guidelines the Zero Robotics team intends to follow during the competition. Keep in mind that as in any refereed competition, additional real-time judgments may be required. Please respect these decisions and consider them final.

Above all, the final competition is a demonstration of all the hard work teams have put forward to make it to the ISS. The ZR staff's highest priority will be making sure every team has a chance to run on the satellites. It is also expected that the competition will have several "Loss of Signal" (LOS) periods where the live feed will be unavailable. We will attempt to make sure all teams get to see a live match of their code, but finishing the competition will take priority.

To summarize, time priority will be allocated to:

- 1) Running all submissions aboard the ISS at least once
- 2) Completing the tournament bracket
- 3) Running all submissions during live video

We hope to complete the tournament using only results from matches run aboard the ISS, but situations may arise that will force us to rely on other measures such as simulated matches.

3.3.2 Competition Format

A total of 12 teams will compete on ISS during the Middle School ISS Final Competition this year: eleven finalist teams (one team from each participating states) and one wild card team. Selection of the wild card team will be determined by ZR staff. The twelve teams will be divided into 2 conferences for the ISS competition.

Each conference will include 2 brackets of 3 teams each (as shown in Figure 9). Each bracket will play 3 matches in round-robin style: team A vs. B, B vs. C, and C vs. A.

After the round-robins are complete, there will be a winner of each bracket (shown as BR1, BR2 in Figure 9.) The following rules determine the winner:

1. The team with the most wins advances
2. If teams are tied for wins, the team with the highest total score advances
3. If scores are tied, simulation results will be used to break the tie

A single semi-final match between the top 2 bracket winners in each conference will determine the conference winners.

The winning team from each conference will play a single match to determine the Zero Robotics ISS Champion. The losing team will be awarded 2nd place.



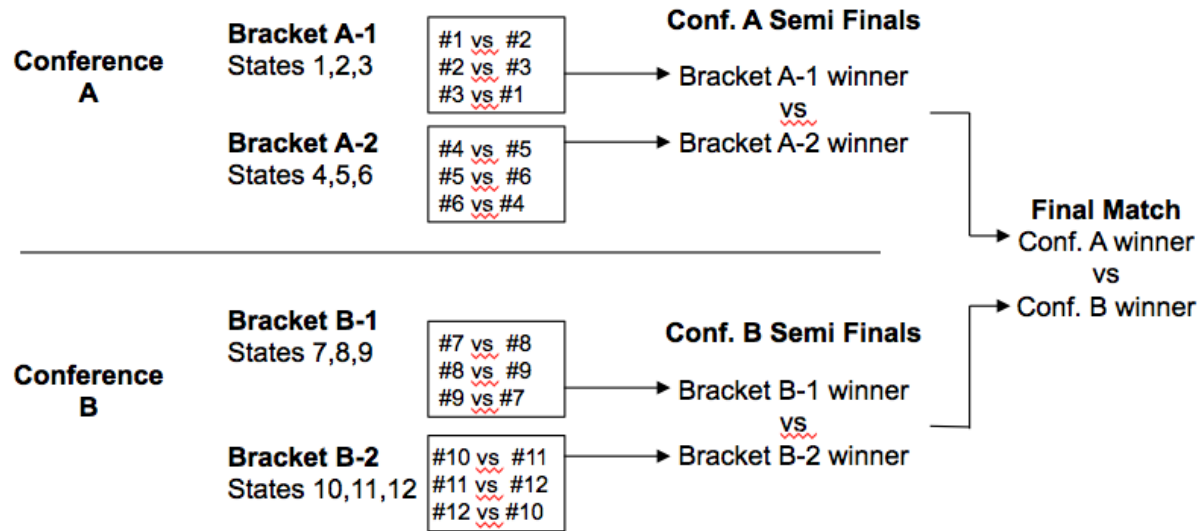


Figure 9: ISS Competition Bracket

Definition: Successful Match

- Both satellites move correctly to initial positions
- Both satellites have normal motion throughout the test
- Both satellites return a valid score
- Neither satellite expends its CO₂ tank during a test run

Definition: Simulated Match

In advance of the competition, the ZR Team will run a simulated round robin competition between all participating teams. The results from matches in this competition will be used in place of ISS tests if necessary (see below.) The results of a simulated match will only be announced if they are used in the live competition.

3.3.3 Scoring Matches

If the match is successful, the scores will be recorded as the official score for the match. If the first run of a match is not successful, the match will be re-run, time permitting. If the second run of a match is not successful, the results from a simulated match will be used.

4 Season Rules

4.1 Tournament Rules

All participants in the Zero Robotics Middle School Summer Program 2015 must abide by these tournament rules:

- The Zero Robotics team (MIT / ILC / Aurora) can use/reproduce/publish any submitted code.
- In the event of a contradiction between the intent of the game and the behavior of the game, MIT will clarify the rule and change the manual or code accordingly to keep the intent.
- Teams are expected to report all bugs as soon as they are found.
 - A “bug” is defined as a contradiction between the intent of the game and actual behavior of the game.
 - The intent of the game shall override the behavior of any bugs up to code freeze.
 - Teams should report bugs through the online support tools. ZR reserves the right to post any bug

reports to the public forums (If necessary, ZR will work with the submitting team to ensure that no team strategies are revealed).

- Code and manual freeze will be in effect 3 days before the submission deadline of a competition. During this time, no changes will be made to either the game code or the manual.
 - Within the code freeze period the code shall override all other materials, including the manual and intent.
 - There will be no bug fixes during the code freeze period. All bug fixes must take place before the code freeze or after the competition.
 - The code is finalized at the ISS Final Competition freeze (unless there is a critical issue which will affect the final tournament, including lessons learned from ground hardware testing and simulation.)
- Game challenge additions and announcement of TBA values in the game manual may be based on lessons learned from earlier parts of the tournament.

4.2 Ethics Code


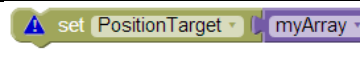
- The ZR team will work diligently upon report of any unethical situation, on a case by case basis.
- Teams are strongly encouraged to report bugs as soon as they are found; intentional abuse of an un-reported bug may be considered as unethical behavior.
- Teams shall not intentionally manipulate the scoring methods to change rankings.
- Teams shall not attempt to gain access to restricted ZR information.
- We encourage the use of public forums and allow the use of private methods for communication.
- Vulgar or offensive language, harassment of other users, and intentional annoyances are not permitted on the Zero Robotics website.
- Code submitted to a competition must be written only by students.

5 ZR User API

The following reference table explains how to use common API and game functions for the CosmoSPHERES game.

SPHERES Controls API Functions*

All of the SPHERES Controls API Functions except DEBUG are accessed as members of the API object; that is they are called as `api.functionName(Arguments)`.

Name	Description	
Void setPos(float x, float y, float z);	Moves the player's satellite to the given x, y, and z coordinates.	
void setPositionTarget(float posTarget[3])	Moves the player's satellite to a point of your choice. You can select a point by creating a three element array, where each element represents an x, y, or z coordinate.	

void setAttitudeTarget(float attTarget[3])	Rotates the player's satellite to face along the x, y, or z axis. You can select the direction by creating a three element array, where each element represents the x, y, or z unit vector of the direction you want to face. For more information, see the <i>More Simple Arrays and setAttitudeTarget Function</i> tutorial on the ZeroRobotics website.	
void getMyZRState(float myState[12])	Retrieves the state of your SPHERE (location, velocity, attitude, and angular velocity). The state will be stored in a twelve element array that you create beforehand. After calling this function, the first three elements of your array will hold the x, y, and z coordinate of your SPHERE's location; the next three elements will hold the x, y, and z components of the velocity; the next three elements will hold the x, y, and z components of the attitude vector; and the final three elements will hold the x, y, and z components of the angular velocity.	
void getOtherZRState(float otherState[12])	Same as getMyZRState but gets the state of the opponent's satellite.	
unsigned int getTime()	Returns the time (in seconds) elapsed since the beginning of the game.	
DEBUG(("Some text!"))	Prints the supplied text to the console. If you are coding in the text editor, do not type api. before this function and make sure to use double parenthesis.	

*DEBUG is found in the Debug section, not SPHERES Controls, and it is not an API function


CoronaSPHERES Game Functions- Actions

CoronaSPHERES Game functions are called as members of the game object, that is, game.functionName(Arguments)

NOTE: **POI Id** is the number assigned to a particular Point of Interest (see Game Manual for details)

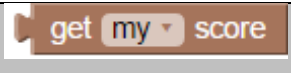
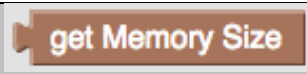
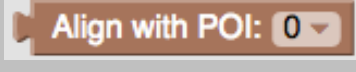
Name	Description	
void takePic(int poiID)	Takes a picture of the Point of Interest (POI) with the id poiID and stores it in the satellite's memory. The camera will not work if there are no available memory slots.	
void uploadPic()	Uploads pictures from the satellite to earth and disables the camera for 3 seconds.	



<p>void turnOn() void turnOff()</p>	<p>Turn satellite on or off.</p> <p>Turn on: Begins process to turn satellite on. Takes 5 seconds. (unless game just beginning)</p> <p>Turn off: Protects from solar flare. Satellite will drift.</p>	

CoronaSPHERES Game Functions- Information

CoronaSPHERES Game functions are called as members of the game object, that is, game.functionName(Arguments)

Name	Description	
int getNextFlare()	This function will return the number of seconds until the next solar flare up to 30 seconds before the flare occurs. If this function is called outside this 30 second window (including anytime during the flare), it will return -1.	
float getFuelRemaining()	Returns remaining fuel.	
float getPlayerScore(int Player)	Returns player's score if "my" (Player =0) is selected. Returns opponent's score if "other's" (Player =1) is selected.	
bool hasMemoryPack(int playerId, int packID)	Returns true if specified player has specified memory pack (0 or 1).	
int getMemorySize()	Returns the current total number of pictures the satellite can store.	
int getMemoryFilled()	Returns the number of valid pictures currently saved in the camera.	
bool alignLine(int poiID)	Returns true if the SPHERE is facing the poi whose ID is given. It does not check if poi is in field of view or if sphere is in the correct zone.	
void getPOILoc(float position[3], int id)	Stores the location of the POI with the id number id in a three element array of your choice. After calling this function, each index in your array will hold the x, y, or z coordinate of the POI's location.	

API Note Index



API Note 1: *the function takePic(int poiID) will not work under certain conditions. If the current number of pictures saved in the camera is equal to the maximum number of pictures the satellite can store (getMemoryFilled() = getMemorySize()), the camera will not be able to take and store anymore pictures. Also, the camera is disabled for 3 seconds following each time takePic(int poiID) is called and the camera is disabled for 3 seconds following each time uploadPic() is called.*

Hint: once your memory pack is full, upload your pictures to free up room.

API Note 2: *the function getNextFlare() will start returning the time until the next solar flare when a solar flare is coming up soon.*

Hint: If you call this function often, you will be alerted of a solar flare before it happens and you can allow yourself enough time to get in the shadow zone, for example. If you are not in the shadow zone during a solar flare, the solar flare may damage your satellite and you will lose points and lose any pictures not yet uploaded.

6 Lists of Figures and Tables

6.1 List of Figures

Figure 1 Game Overview.....	4
Figure 2: How to Check Project Code Size	6
Figure 3 POI Locations.....	7
Figure 4: Picture Taking Tolerances (diagrams not to scale)	9
Figure 5: Memory Pack Locations.....	10
Figure 6 Solar Flare	11
Figure 7 Asteroid Collision	12
Figure 8 Maneuver to Collect Item.....	13
Figure 9: ISS Competition Bracket.....	17

6.2 List of Tables

Table 1 Interaction Zone Dimensions.....	5
Table 2 Zone Radii Positions.....	5
Table 3 Shadow Zone Dimensions	5
Table 4 Fuel Allocation	6
Table 5 SPHERES Satellite Deployment Locations.....	6
Table 6 Asteroid and SPHERES Measurements	7
Table 7 POI Locations	7
Table 8 Picture Values in Inner and Outer Zones.....	8
Table 9 Max Angle allowed (in degrees).....	8
Table 10 Memory Upgrade Pack Locations	10
Table 11 Number of Solar Flares.....	10
Table 12 Memory Upgrade Pack Locations (repeated)	13



Table 13 Point Values..... 14

Table 14 Tournament Key Dates 15

7 Revision History

Revision	Date	Changes	By
1.0	5/19/15	Initial release (review API table formatting, list of figures, TOC, etc)	zerorobotics@mit.edu
1.1	6/4/15	Revision One (corrections and clarifications of rules to correspond with source code, typo corrections, format corrections)	UROPS
1.2	7/1/15	Revision Two: Error in section 1.3: SPHERES start on the Y axis not on the X axis	UROPS

