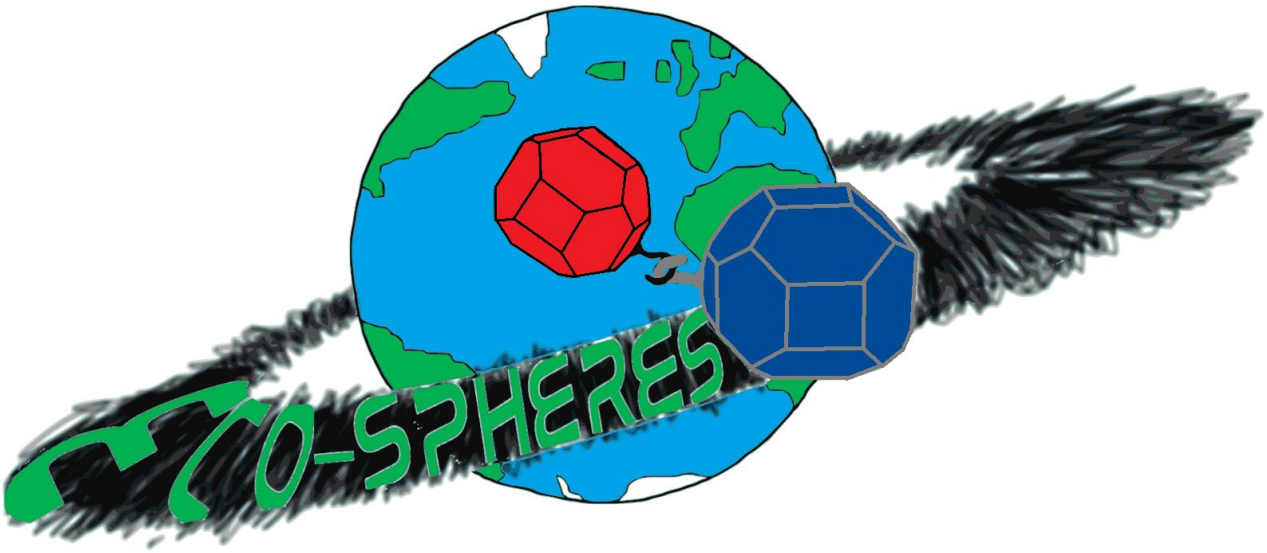
ZERO ROBOTICS

HIGH SCHOOL 2018

GAME MANUAL

Version: 2D-0.9.1 (Preview Release)



ECO-SPHERES

Evade, Capture, deOrbit space debris Spheres Program

TO: ZERO ROBOTICS HEADQUARTERS

INCOMING SOS

The following transmission was received just moments ago:

> *SOS*
> *RED SPHERES_15569824*
> *SOS*
> *Thruster_09 .... UNRESPONSIVE*
> *Thruster_12 ... UNRESPONSIVE*
> *Location_Data ... UNKNOWN*
> *Radar ... NULL*
> *Target_Location ... LEO SATELLITE 92F003 RETRIEVAL*
> *Last_Target ... GEO SATELLITE 35B117 RETRIEVAL*
> *SOS*
> *RED SPHERES_15569824*
> *SOS*

The Red SPHERES Satellite is in trouble.

Increasing numbers of satellites are being deployed to Low Earth Orbit (LEO) to study Earth's atmosphere, climate, land, oceans, and weather. To protect the success of this important research the SPHERES program is working with space agencies internationally to identify and remove space debris from LEO. Recently the SPHERES program deployed ECO-SPHERES as a part of its Evade, Capture, de-Orbit (ECO) initiative to remove debris. The ECO-SPHERES design includes a hook for the SPHERES to use to tow any type of cargo.

By its very nature, capturing and removing junk puts expensive debris removal satellites in harm's way. Any clean-up method is prone to damage. As you heard in the transmission above, one of our SPHERES Satellites has been damaged by debris and we need to retrieve it. The mission ahead of us, however, is dangerous. We must deploy another ECO-SPHERES to traverse the crowded LEO and, instead of moving debris, it needs to hook onto the damaged red SPHERES Satellite and bring it back to safety.

Program your satellite to avoid debris while trying to locate the red SPHERES. Use expert geometry to latch onto the damaged satellite, and bring it back to safety, taking into account the momentum of your precious cargo as to not further damage it.

Good luck to all participating space engineers.

Alvar Saenz-Otero
SPHERES Lead

Table of Contents

1 Game Overview

Each team will compete individually by programming one SPHERES ("player SPHERES") to interact with a pre-programmed SPHERES ("target SPHERES"). The game consists of four phases: Debris Field Navigation, Rendezvous, Hooking, and Towing/Return Target.

Debris Field Navigation

The player SPHERES must navigate through a field of debris, thruster damage will be incurred for collisions with debris.

Rendezvous

The player SPHERES must approach the target SPHERES and meet the "rendezvous conditions." Once "rendezvous conditions" are met, all Space Debris will cease to exist and the target SPHERES will stop moving and actively hold position with its thrusters.

Hooking

The player SPHERES will then attempt to capture the target SPHERES with its hook.

Towing/Return Target

The player SPHERES must then get the target SPHERES back to the "safe zone".

Each team will compete to have the most points when the game time is up. Each game lasts 210 seconds.

2 Game Rules/Phases of Game Play

In order to be victorious over other participating teams, each player SPHERES should carefully navigate the debris field to rendezvous and hook the target SPHERES and return it to safety all while managing fuel, time, and their location on the gameplay area.

Note: Satellite position relative to all game features is determined by the location of the center of the satellite as stored in the state variables (ZR state or SPHERES state) unless indicated otherwise.

2.1 Playing Field

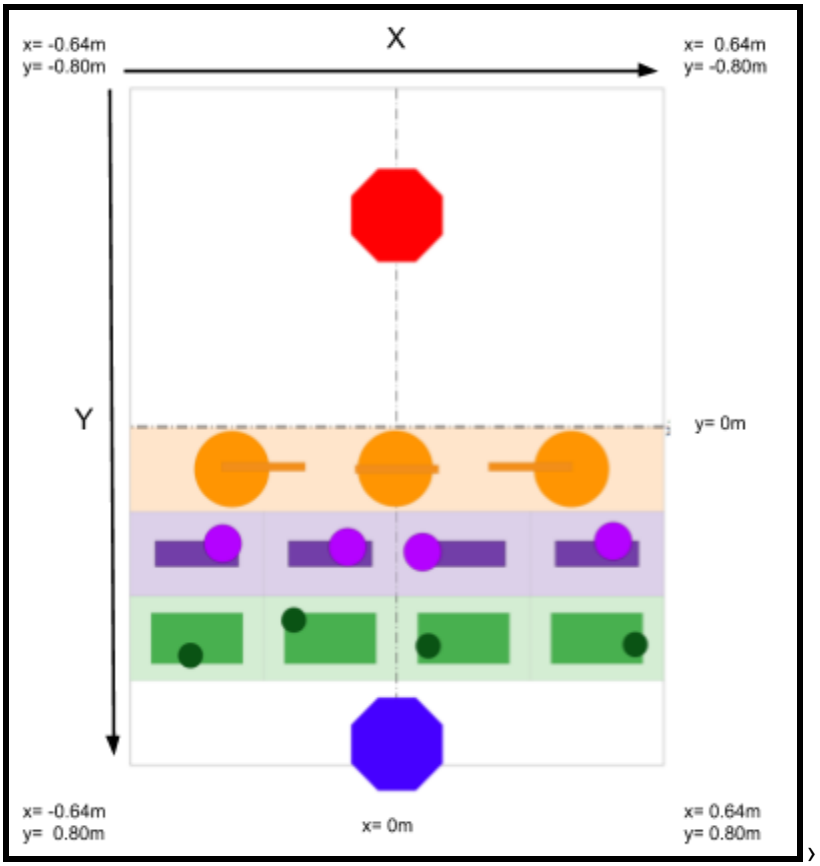
The playing field is determined by the size of the area available for the SPHERES satellites to move inside the International Space Station. The simulation creates an *interaction zone* of the same size. If players leave the Interaction Zone, they are considered out of bounds.

The Interaction Zone for the game has the following dimensions:

Interaction Zone Dimensions

	2D	3D	Alliance
X [m]	*[-0.64 : +0.64]*	*TBA*	*TBA*
Y [m]	*[-0.80 : +0.80]*	*TBA*	*TBA*
Z [m]	*n/a*	*TBA*	*TBA*

A player will be penalized a portion of its fuel allocation for every second it remains out of these bounds.



Playing Field Area and Interaction Zone Dimensions

The Playing Field Area and Interaction Zone Dimensions are shown in the Figure above. The Debris Field is represented by the green, purple and orange circles. At the start of the ECO-SPHERES game, the Player SPHERES (Blue SPHERES) is on one side of the Debris Field and the Target SPHERES (Red Spheres) is on the other side of the Debris Field.

2.1.1 Initial Position

The Blue and Red SPHERES satellites are deployed to the same initial position every game as follows:

Initial Positions

	2D	*3D*	*Alliance*
Player SPHERE (Blue)			
X [m]	*0.0*		
Y [m]	*0.75*		
Z [m]	*0.0*		
Target SPHERE (Red)			
X [m]	*0.0*		
Y [m]	*-0.5*		
Z [m]	*0.0*		

The satellite radius is 0.11m.

2.1.2 SPHERES "Pointing Face" for ECO-SPHERES

Since the hooks used in the ECO-SPHERES game mount on the +X face (docking face) of the SPHERES both satellites are configured with the +X face as the pointing face for the ECO-SPHERES game. This means that the `setAttitudeTarget()` function will determine the pointing direction of the hook of the satellite on the +X face.

(This is as compared to Free Mode which assigns the -X face of the SPHERES as the pointing face when using `setAttitudeTarget()`)

2.2 Debris Field Navigation

During the first phase of ECO-SPHERES, the player SPHERES must carefully navigate through a Debris Field in order to reach the Target SPHERES. Collisions with debris will result in thruster damage.

2.2.1 Bounds of Debris Field

Space Debris obstacles are located in range 0m < y < 0.7m

Parameter	Description	Value (m)
$y_{debris,start}$	*y value of start of debris field*	*0.7*
$y_{debris,end}$	*y value of end of debris field*	*0.0*

2.2.2 Size and Position of Space Debris

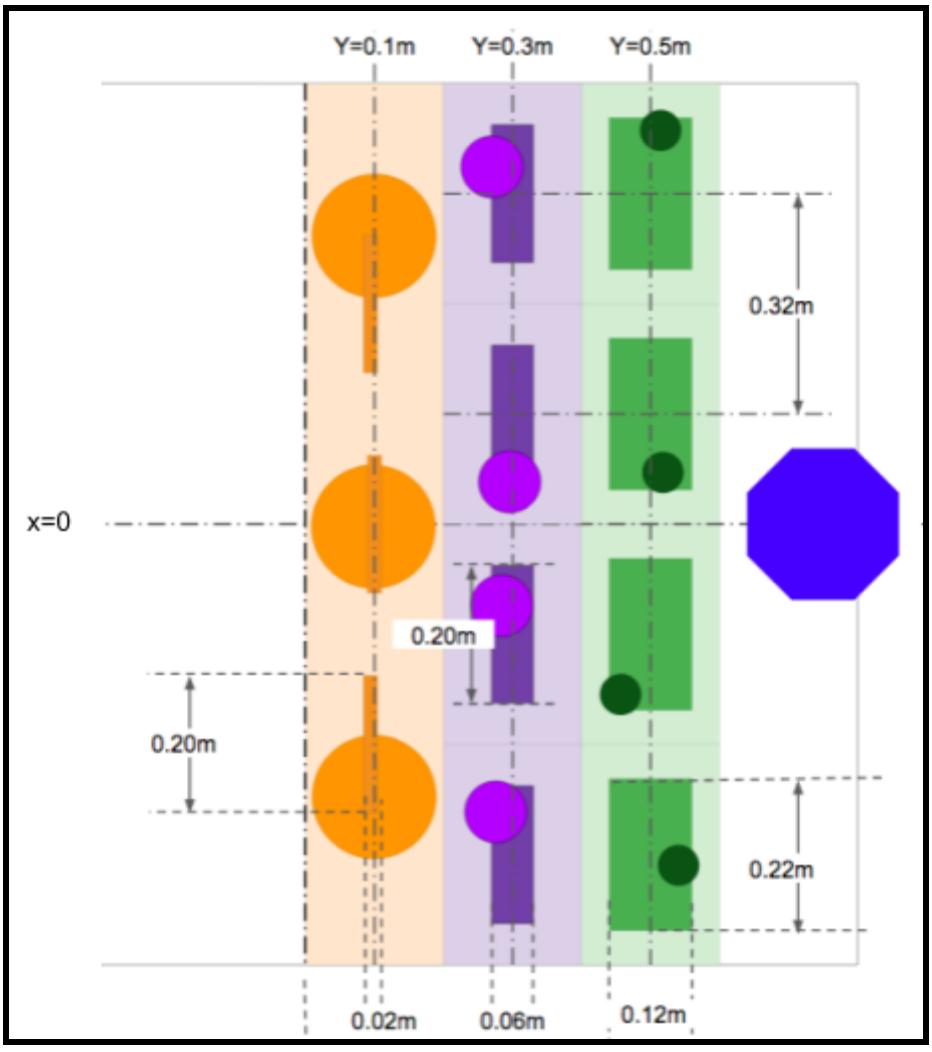
In the 2D game there are eleven Debris obstacles located within the Debris Field. Obstacles occur in the three sizes shown in the table below. Use the Function: `game.getDebris` to retrieve information about debris position and size.

Debris Type	radius(m)
Small	*0.03*
Medium	*.045*
Large	*.09*

Space Debris Sizes

Game	Number
2D	*11*
3D	*TBA*
Alliance	*TBA*

Number of Debris



Possible locations of Space Debris within Debris Field
(represented by orange, purple, green rectangles)

The three different sizes of debris occur in 3 zones centered on y=0.1m (large debris), y=0.3m (medium debris) and y=0.5m (small debris) as shown in the figure above. The orange, purple

and green rectangles (with specified width and length) in the figure above reflect the possible locations for debris within each zone. The debris position is specified by the location of the debris center.

2.2.3 Collision with Space Debris

Collision with debris results in thruster damage, which reduces the maximum translational and rotational velocity of the SPHERES. The damage incurred depends on the size of the debris as shown in the table below. If the player SPHERE is out of bounds and it crosses into any of the debris zone areas (small: 0.60m < Y < 0.40m; medium 0.40m < Y < 0.20m; large 0.20m < Y < 0.00m) this will incur thruster damage equivalent to collision with one of the Debris located in that region.

Debris Collision Damage			
Debris Type	**2D**	**3D**	**Alliance**
Effect	***Decrease in total thrust power (position and attitude) as a % of original thrust.***	***TBA***	***TBA***
Small	***10% (TBR)***	***TBA***	***TBA***
Medium	***25% (TBR)***	***TBA***	***TBA***
Large	***50% (TBR)***	***TBA***	***TBA***

2.2.4 Cancel Debris Collision Damage tool (for test only)

In order to facilitate concurrent development of code for all phases of the ECO-SPHERES game a flag has been added to the simulation window which can be set to allow the player satellite to pass through the debris field without thruster damage. Look for the “Cancel Debris” field at the bottom of the simulation window, which can be toggled between 0 and 1.

0 = incurs normal thruster damage (default)
1 = cancels debris collision damage.

2.3 Rendezvous

During the second phase of the game the player SPHERES must rendezvous with the Target SPHERES.

2.3.1 Motion of Target SPHERES

In the 2D game, the target SPHERES will be following a specific translational and rotational pattern. The player SPHERES must account for the Target SPHERES motion in order to achieve successful Rendezvous.

2.3.2 Criteria for Successful Rendezvous

The "rendezvous position" will be defined by the following conditions being met: 1) the center to center separation of the SPHERES must be less than ε_r, 2) the hook of the player satellite must be pointing at the center of the target satellite within $\varepsilon_{r,\theta}$, 3) the player satellite must be moving with a speed (absolute value, not directional velocity) less than v_max, and 4) the player satellite must have a body rotation rate under w_max. Note that the two velocities are inertial, and not relative to the target satellite.

Parameter	Description	Value	Units
ε_r	*max "center to center" distance for rendezvous*	*0.40 (TBR)*	*m*
$\varepsilon_{r,\theta}$	*max angular tolerance for rendezvous*	*0.052 (TBR)* *3*	*rad* *deg*
v_max	*maximum local speed of the player satellite*	*0.005 (TBR)*	*m/s*
w_max	*maximum body rotation rate of the player satellite*	*0.0175 (TBR)* *1*	*rad/s* *deg/s*

Note that the Rendezvous criteria only checks that the player is pointing correctly to the target. However, successful hooking will also require the target to point correctly to the player. Because MIT controls the motion of the target, there is no requirement on the pointing of the target to the player. However, there is a scoring bonus if the target hook is pointing to the player or penalty if the target hook is pointing away from the player. The scoring section details this bonus/penalty.

2.4 Hooking

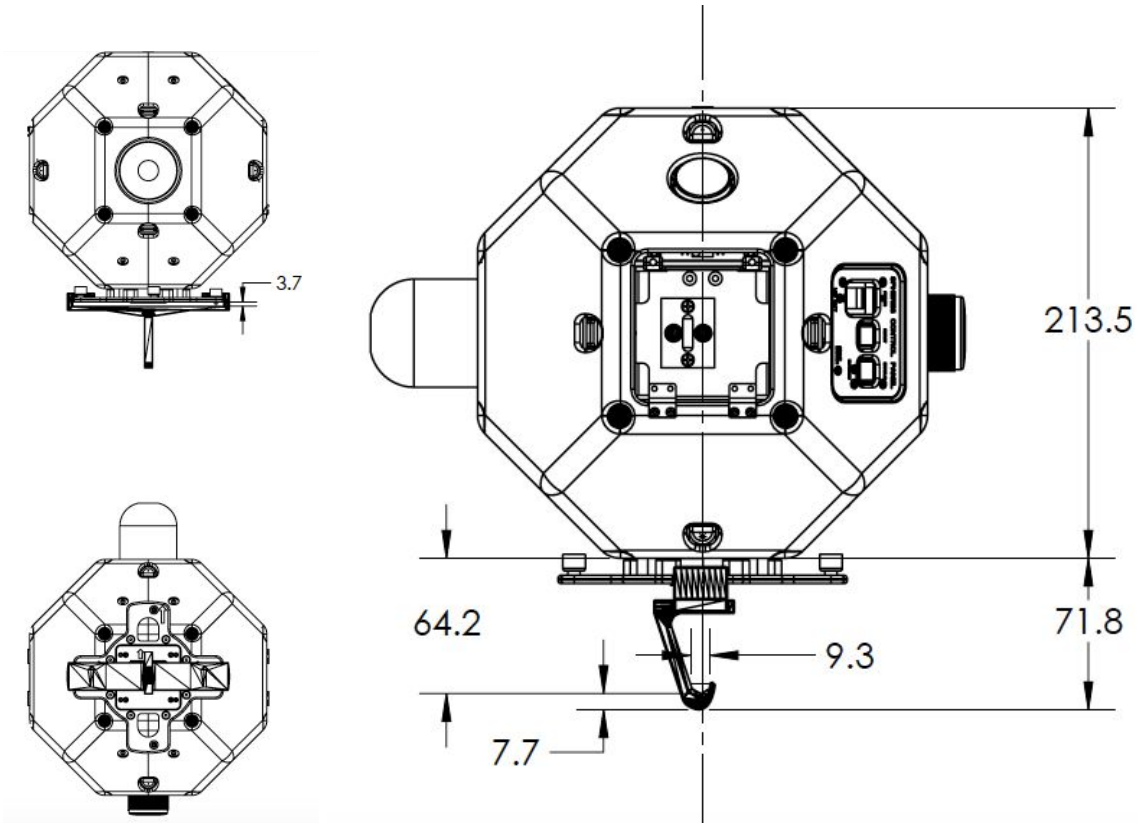
During the third phase of the game the player SPHERES must successfully Hook the Target SPHERES.

2.4.1 Position and Motion of Target SPHERES during Hooking

Once the rendezvous phase has been completed, all Space Debris will cease to exist and the target SPHERES will come to a stop as quickly as possible and actively hold its position. It will also orient in a predetermined direction to facilitate hooking.

2.4.2.Hooking Criteria

The Player SPHERES must carefully position its hook into the hooking area of the Target SPHERES. The figure below shows the 3D printed hook assembled on the SPHERES. The Hook assembly attaches to the SPHERES docking face (+x face).



Dimensions shown are mm.

Key Details/Dimensions	Value
Hook assembly attaches to the SPHERES docking face (+x face)	*N/A*
Distance between center of SPHERES and hooking center	*0.17095m*
Width of Hook	*.008m*

2.5 Towing/Return of Target SPHERES to “Safe Zone”

The goal of the fourth and final phase of the game is for the Player SPHERES to return the Target SPHERES back to a “Safe Zone”, ie, the positive Y (+*Y*) half of the playing field.

2.6 Scoring

Scoring includes both Time Dependent Scoring and Bonus Point Scoring. The function `game.getScore` can be used to check player score.

2.6.1 Time Dependent Scoring

The primary way teams earn points is by completing different phases of the game: Debris Field Navigation, Rendezvous, Hooking, Return to Safe Zone.

Each phase has a baseline point value earned when the phase is completed. Additionally, the faster teams complete each phase, the more points they will earn.

Phase	Time	Min Score	Max Score
Debris Field	30s	0.25	0.50
Rendezvous	60s	0.25	0.50
Hooking	90s	0.50	1.00
Return Target	30s	0.50	1.00

Time Dependent Scoring

The number of points awarded for each stage is determined by:

Symbol	Description
$p_{0,L}$	Minimum points awarded for crossing debris field
$p_{0,U}$	Points awarded for crossing debris field in 0 time
$\Delta T_{0,ref}$	Time for crossing debris field which causes minimum score
$p_{1,L}$	Minimum points awarded for achieving rendezvous
$p_{1,U}$	Points awarded for achieving rendezvous in 0 time

$\Delta T_{1,ref}$	Time for achieving rendezvous which causes minimum score
$p_{2,L}$	Minimum points awarded for achieving hooking
$p_{2,U}$	Points awarded for achieving hooking in 0 time
$\Delta T_{2,ref}$	Time for achieving hooking which causes minimum score
$p_{3,L}$	Minimum points awarded for returning target SPHERES
$p_{3,U}$	Points awarded for returning target SPHERES in 0 time
$\Delta T_{3,ref}$	Time for returning target SPHERES which causes minimum score

$$pn,L + (pn,U - \frac{pn,U-pn,L}{\Delta T\,n,ref}\ \Delta T\ \text{n})(\Delta T\ \text{n} < \Delta T\ \text{n,ref})$$

where pn,L is the baseline score earned when phase n is complete, pn,U is the maximum possible score that could be earned (if the phase was completed in zero time), ΔTn is the time elapsed since the phase began, and ΔTn,ref is the time at which no more than the min score will be awarded for that phase.

2.6.2 Bonus Points

2.6.2.1 Rendezvous Pointing Bonus Points

At the end of the rendezvous phase (when the player calls `game.completeRendezvous` successfully) a bonus or penalty will be scored based on how close the hook of the **target** is pointing to the center of the **player**.

The bonus/penalty is assessed as follows:

- Determine the "pointing vector" between the target to the player
- Take the *dot product* of the *normalized* "pointing vector" and the ZR attitude of the target (which is itself a normalized pointing vector)
- Square the *dot product* magnitude (which increases the effect of misalignment)
- Scale by 2.5 (so 2.5 is the maximum bonus and -2.5 the maximum penalty)

Symbol	Description	Max bonus	Max penalty
b_r	Rendezvous pointing bonus	2.5	-2.5

Note: The *dot product* of two *normalized* vectors returns 1 if the vectors are perfectly aligned, 0 if they are 90° offset, -1 if they are pointing away from each other, and a fraction less than 1 when they are partially aligned.

Note: the sign is maintained, only the magnitude is squared.

2.6.2.2 Fuel Bonus Points

Once the SPHERES return to the "Safe Zone", additional points will be awarded for any remaining fuel. More weight will be given to remaining fuel in the players SPHERES than in the target SPHERES ($f_{player} > f_{target}$) The additional fuel points will be calculated by multiplying the percentage of remaining fuel (expressed as a decimal) by the fuel weights as follows: $\omega_{player} * f_{player} + \omega_{target} * f_{target}$.

Symbol	Description	Value
ω_{player}	Percentage of remaining player SPHERES fuel	From game
ω_{target}	Percentage of remaining target SPHERES fuel	From game
f_{player}	Multiplier for player SPHERES fuel	TBA
f_{target}	Multiplier for target SPHERES fuel	TBA

2.6.3 Scoring Equation

Given the above information, the total score at the end of the game is calculated as follows:

$$Score = \sum_{n=1}^{4} [pn,L + (pn,U - \frac{pn,U-pn,L}{\Delta Tn,ref} \Delta T\,n)(\Delta T\,n < \Delta T\,n,ref)] + (\omega_{player} * f_{player} + \omega_{target} * f_{target}) + b_r$$

2.7 End of game

The game ends after 210 seconds. Final Score is calculated.

2.8 SPHERES Satellites

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. The actual SPHERES satellites aboard the ISS, like any other spacecraft, have limited fuel (in this case liquid carbon dioxide), power (in this case AA battery packs), and computer (a digital-signal processor). These resources are limited and must be used wisely. Therefore, the players of Zero Robotics are limited in the use of these resourced by virtual limits

within the game. The use of batteries is limited by having a fixed game time of 210 seconds. The other limitations are detailed below.

The *nominal* properties of the SPHERES satellites are summarized as follows:

Property	Value	Units
Radius	*0.11*	*m*
Diameter	*0.22*	*m*
Mass	*4.0*	*kg*
Single Thruster Force	*0.11*	*N*

These are called *nominal* properties because they are not exact. The Mass changes up to 0.2kg as fuel is consumed. The single thruster force is affected by how many thrusters are open at one time, varying up to 20% of the nominal force. In addition, while every attempt was made to align the thrusters with the satellite body axes, there are imperfections in the alignment (within 2°), therefore not all the force goes in the exact desired direction.

2.8.1 Fuel

Each player is assigned a virtual fuel allocation of 60 seconds of total accumulated thruster firing time. This is calculated by summing individual thruster firing during the game. Once the allocation is consumed, the satellite will not respond to the player SPHERES control commands. It will fire thrusters only to avoid leaving the Interaction Zone or colliding with the other satellite.

Any action that requires firing the thrusters (rotating, accelerating, decelerating), whether it was commanded by the player, due to activate collision avoidance or out-of-bounds breaking, or other penalties of the game play, will consume virtual fuel allocation.

The function `getFuelRemaining` can be called to obtain the fraction of fuel still available during a game. The function returns 1.00 for a full tank, and then a fractional value (e.g. 0.50 = half a tank) until reaching 0.0. Once the function returns 0.0 all user motion commands will be ignored.

2.8.2 Collision Avoidance

Collision Avoidance is not activated for this game, since part of the goal of this year's game is to get close enough to hook. This means that in the simulation the satellite **will** be able to go across each other, and the simulation will **not** stop what would be real collisions aboard ISS. It is up to each team/alliance that advances to the ISS finals that their code will not result in actual physical collisions (as much as possible).

2.8.3 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 the “Code Size Estimate” menu option, 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.

2.8.4 Noise

It is important to note that the SPHERES simulations create noise similar to that experienced by the satellites aboard the ISS. This noise means two main things:

- The satellites will never know exactly where they are; their “estimate” of their location reflects that the sensors are not perfect, so at all times even if the satellite is supposed to be completely still, their location will vary approximately ±0.005m independently on every axis.
- The satellites will not thrust perfectly. While the thrusters use identical designs, each thruster has a slight variation in thrust, and the total thrust varies by the number of thrusters used at one time. This means that the thrust of the satellites may vary in general 10% and up to 20% in some cases.

This is fully intended as part of the challenge and reflects uncertainties in real aerospace engineering systems, which have imperfect dynamic models, sensors, and actuators. The best performing solutions will be those that prove to be robust to these variations and a wide variety of different initial conditions of the gameplay features.

2.9 ISS Finals: research with Zero Robotics

The teams that participate in the ZR HS2018 ISS Finals will be conducting “hooking in space” for the first time ***ever*** aboard the ISS with SPHERES. Compared with previous finals, the ISS Finals will **not** be a continuation of exactly the same game as that done in simulation:

The ZR HS2018 ISS Finals will feature only the hooking and towing phases.

The Debris phase will **not** exist in the ISS Finals, there will be no “random” feature to the finals - the ability to “hook” will be the priority, as it is already a hard enough problem. Further, while the teams will need to “rendezvous”, there will be no game features regarding that phase. Rather, the teams will be responsible to correctly align themselves for hooking, without a check by the game.

However, the ISS Finals will have an aspect of the “debris” phase of the game:

- Each Alliance will start with the Thruster Health as determined by the average of their 10 games that they qualified for ISS.

The ISS Finals will try to increase the complexity of the hooking, but that is TBR. The initial idea is:

- First round: Target is not moving
- Second round: Target is either translating or rotating (TBR)[1]
- Final round: Target is both translating and rotating (TBR)[1]

During the ISS Finals the “points” to advance in rounds will be based solely on the hooking and towing parts of the original game, following the same point structure (time-based ability to hook and then to tow).

2.9.1 Different Games

In order to help teams practice for the ISS Finals, there will be **two** games available during the full tournament:

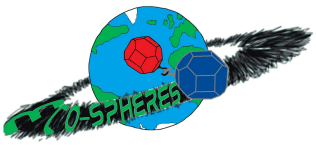
- Complete game - used for elimination rounds
- Hook game - game without debris and with the ability to skip rendezvous motion, not used for elimination rounds, only to practice rendezvous, hooking, and towing.

Because the “Debris” field phase is both challenging and can use more complex algorithms, and because it will **not** be part of the ISS finals (therefore not actually limited by the SPHERES hardware), the Code Size will be increased for the elimination rounds. However, the ISS Finalists will need to keep their code to within the limits of that supported by the actual hardware.

Program	Elimination	Code Size
Complete Game	*Yes*	*10x*
Hook Game	*No, ISS Only*	*1x*

[1] If the target is moving in any way, it will NOT stop rotating upon rendezvous, as there will be no function to indicate so; players will need to hook to a moving target in that case.


SPHERES


COSSPHERES

3 Game API (TBR)

The following table lists the functions available in this year's game. In order to use these functions, use the syntax `game.functionName(inputs)`, for example:

```
game.getDebris()
```

For the general Zero Robotics functions use the syntax `api.functionName(inputs)`, for example:

```
api.setPositionTarget(posTarget)
```

(unless they are math functions, which can be called without reference to the instance).

The generic ZR user API (in both C and MATLAB languages) is available at:
http://static.zerorobotics.mit.edu/docs/tutorials/ZR_user_API_2017.pdf

ECO-SPHERES API Reference

Name	Description
C: `void getDebris(float debris[NUM_DEBRIS][4])` MATLAB: *pending*	Puts the following information into the array "debris": [debris[n][0], debris[n][1], debris[n][2]] = position of the debris Debris[n][3] = radius of the debris The array must be of length "NUM_DEBRIS", which is: 2D Game: 11 3D Game: TBA Alliance Game: TBA It is the responsibility of the players to correctly size the array.
C: `bool checkRendezvous()` MATLAB: *pending*	Returns true if the Rendezvous requirements are met. Returns false otherwise.
C: `bool completeRendezvous()`	If the Rendezvous requirements are met, will complete the phase and command the target satellite to stop moving. The first time this is called successfully is the scoring time for the Rendezvous phase; the pointing of the target satellite to the player at this instant is also used for bonus points. Returns true if the Rendezvous completes successfully (or had already completed in the past); returns false if the requirements are not met.
C: `bool checkHooking()` ***pending*** MATLAB: *pending*	TBA
C: `int getThrusterHealth()` MATLAB: *pending*	Returns the thruster health of the player satellite between 0 to 100 %.
C: `float getScore()` MATLAB: `myScore = getScore()`	Returns player's score

C: `float getFuelRemaining()` MATLAB: `fuel = getFuelRemaining()`	Returns remaining fuel as a fraction of total fuel allowed (1.00 = full tank; 0.50 = 50% remaining; 0.00 = empty tank).

4 The Leaderboard for ECO-SPHERES

4.1 Introduction

This year's leaderboard has been modified to support single player games.

The Leaderboard calculates rankings daily from the beginning of a competition until the submission deadline. Results from prior days have no impact on current days rankings. Only the final standings on the Leaderboard at the end of each competition phase will determine which teams advance to the next phase.

4.2 Daily Rankings

Each day, at 21:59:59 UTC, (except on competition deadlines where posted times apply) your most recently submitted code is collected by an automated system and played as the Blue SPHERES. Each team's code is played a total of ten (10) times. Scores from the 10 games are averaged to determine rankings. Results are posted on the website after rankings are complete. While the daily competition is ongoing the leaderboard will not be visible and clicking on the Leaderboard will return the message: "Results will be posted once the daily leaderboard run is complete". Multiple games are completed to provide teams with more game data given the random elements in the game. Ranking data is refreshed daily.

4.3 Elimination Rankings

On the last day of each competition period, each team's code will be run (10 + n) times until the ranking stops changing or n=15 (every team has played 25 games), whichever comes first. Each iteration every teams' top 10 scores (only) will be averaged to determine rank.

4.4 Tips

The team with the highest average score will be rank 1 on the leaderboard. The best way to improve your rank is the most logical way: keep working at your algorithms. There are no surefire alternatives. Also, don't let fear of a bad game keep you from submitting early. Results of past competitions demonstrate that teams that make many submissions tend to perform better.

Finally, it is important to note that ranking data is refreshed and teams start from scratch daily.

5 Tournament Structure

5.1 2D Practice Simulation Competition

All teams that complete a valid registration are eligible to participate in the 2D practice simulation competition.

Note: Several International teams from non-ISS nations may participate in the 2D and 3D phases of the tournament, however, they will not continue into the alliance phase. These teams will participate by invitation only.

5.2 3D Simulation Competition

All USA, Russian, and Australian teams that complete a valid registration and submit code that ***achieves a positive nonzero score*** by the 2D practice competition deadline are eligible to participate in the 3D simulation competition. However, only the 25 highest scoring teams from each ESA member state will continue into the 3D phase.

When the 3D competition starts the game will be updated with new challenges and the corresponding TBA values will be announced.

5.3 Alliance Formation Event

The top ranked 84 teams on the leaderboard at the end of 3D simulation play will form 28 alliances of three (3) teams each. The 28 alliances will work cooperatively to complete the semifinals and, if not eliminated, the finals. Teams ranked below rank 84 will be invited to participate in the Virtual Finals. See “Virtual Finals Simulation Competition” section below.

The ranking of the teams will be determined by the rankings at the close of the 3D leaderboard.

After the 3D leaderboard rankings are announced, advancing teams will have three days to contact each other to discuss their alliance preferences. Any teams that do not wish to continue in the Tournament will have the opportunity to cede their position to the next ranked team.

The Alliance Formation Event will take place on November 3rd. The alliance formation event or “draft day” is an online event during which teams will follow the alliance selection process described below to invite other teams into alliance and/or accept their place within an alliance. At least one representative from each team must participate in the online “draft day” event for the team to continue to the alliance phase. Additional details about the online “draft day” and how to participate will be provided in advance of the event.

Due to the large number of teams involved, the draft will be split into two parts as shown in the below figure. All teams ranked with odd numbers will participate in Draft part 1 and all teams ranked with even numbers will participate in Draft part 2. Each draft will include 42 teams.

Division of Teams for the Drafts

Draft part 1 teams with odd numbered rankings		
1	29	57
3	31	59
5	33	61
7	35	63
9	37	65
11	39	67
13	41	69
15	43	71
17	45	73
19	47	75
21	49	77
23	51	79
25	53	81
27	55	83

Draft part 2 teams with even numbered rankings		
2	30	58
4	32	60
6	34	62
8	36	64
10	38	66
12	40	68
14	42	70
16	44	72
18	46	74
20	48	76
22	50	78
24	52	80
26	54	82
28	56	84

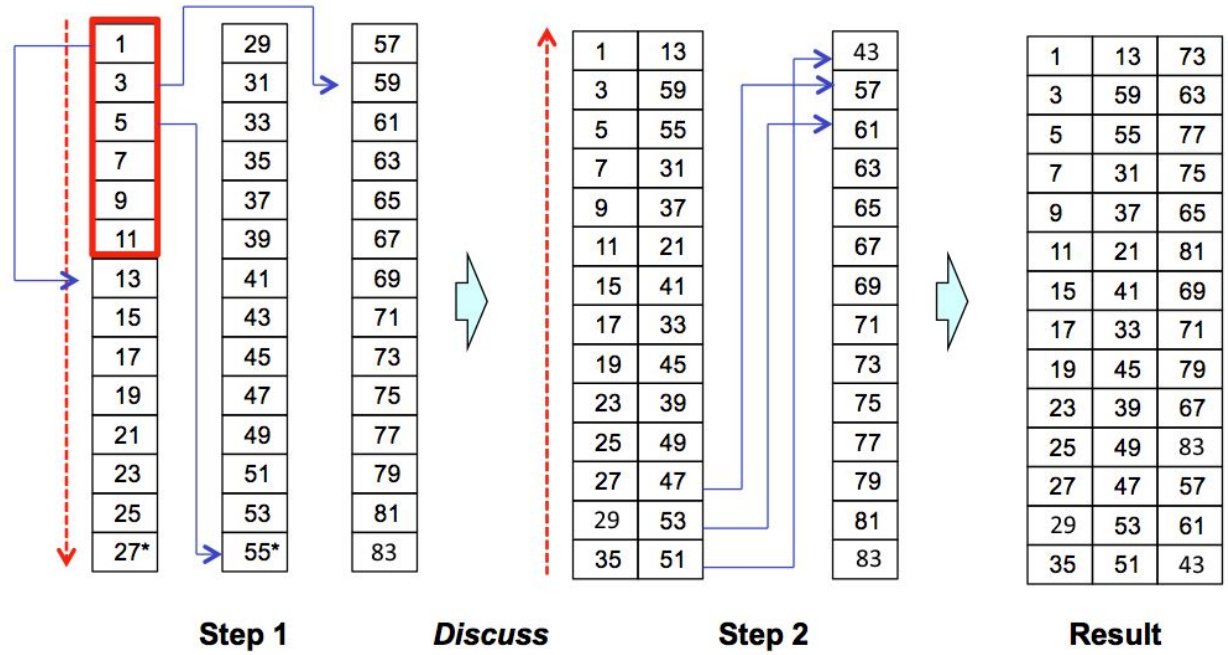
Note: The times of the draft parts are To Be Announced

The alliance selection process will follow a serpentine pattern (illustrated in the Figure below):

- The highest ranked team selects their partner from any except the top 6 ranked teams in their draft.
- The next highest ranked team selects their partner from any of the remaining teams except the top 6 ranked teams in their draft
- 14 pairs are created in this manner
- A break takes place for the new pairs to discuss their selection for the 3rd alliance team
- The “lowest” ranked pair then selects their 3rd team from the remaining 14 teams
- The “2nd lowest” rank pair make the next selection
- Continue until all 14 alliances are formed.
- This process is duplicated in both draft events to end with a total of 28 alliances

Rule: No more than 2 of the teams in an alliance can be from the same continent.

Alliance Creation Process demonstrated for teams with odd number rankings



(*The example shown above is approximate.)

5.4 Semifinal Simulation Competition

The 28 alliances created during the Alliance Formation Event will participate in the semifinal simulation competition.

When the semifinal competition starts, the game will be updated with new challenges and the corresponding TBA values will be announced. These new challenges are intended to be substantial enough to require participation of all alliance teams in preparing competition submissions.

5.5 ISS Final Competition

The top 14 alliances on the leaderboard at the end of semifinal play will advance to the ISS Finals Competition. Alliances ranked below rank 14 will be invited to participate in the Virtual Finals. See “Virtual Finals Simulation Competition” section below for details about the Virtual Finals.

The ISS finals will take place aboard the International Space Station with live transmission to MIT. Teams will be invited to live broadcast events at MIT (US), an ESA location (EU), and at University of Sydney (Australia).

5.5.1 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 game, 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 alliance 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 game with their player, 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 games run aboard the ISS, but situations may arise that will force us to rely on other measures such as simulated games.

5.5.2 Competition Format

The top 14 alliances from Semi Finals and top 2 teams from virtual finals will get to run their code aboard the ISS with the real physical SPHERES satellites and hooks.

There will be three rounds aboard ISS:

Round	Alliances	Task
1	*14 Finalists* *2 Virtual*	*Hook to a still Target*
2	*Top 4*	*Hook to a moving Target (simple)*
3	*Top 2*	*Hook to a moving Target (complex)*

5.5.2.1 Definition: Successful Game

- Both satellites move correctly to initial positions

- Both satellites have normal motion throughout the test
- Satellite returns a valid score
- Neither satellite expends its CO_2 tank during a test run

A game that is not successful will be attempted only one more time.

5.6 Virtual Finals Simulation Competition

All teams participating in the 3D competition that do not advance to the Semi Final competition (Alliance Phase) and all teams participating in the Semi Final Competition that do not advance to ISS Finals will be invited to participate in the Virtual Finals.

The Virtual Finals game will be identical to the Semi Final competition (Alliance Phase) game. Teams participating in the Virtual Finals will submit to a Virtual Finals Leaderboard.

Teams may choose to participate in the Virtual Finals as individual teams or as alliances. Teams that did not participate in the original alliance formation event are welcome to create alliances, if desired. However, once a team/alliance submits code to the Virtual Finals Leaderboard the team/alliance composition cannot be changed.

Once the Virtual Finals Leaderboard closes, the top ranked 2 teams/alliances will advance to the Championship Games of the Virtual Finals. Time permitting, the final Championship Games of the Virtual Finals competition will be competed aboard ISS during the ISS Finals.

6 Season Rules

6.1 Tournament Rules

All participants in the Zero Robotics High School Tournament 2018 must abide by these tournament rules:

- The Zero Robotics team (MIT / Aurora/ ILC) 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 behavior of the game.
    - The intent of the game shall override the behavior of any bugs up to code freeze.

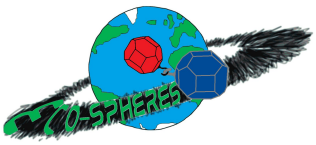
    - Teams should report bugs through the online support tools. ZR reserves the right to post any bug reports to the public forums (If necessary, ZR will work with the submitting team to ensure that no team strategies are revealed).
- Code and manual freeze will be in effect 3 days before the submission deadline of a competition.
- 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.
- Game challenge additions and announcement of TBA values in the game manual may be based on lessons learned from earlier parts of the tournament.

6.2 Ethics Code

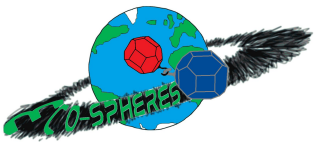
- The ZR team will work diligently upon report of any unethical situation, on a case by case basis.
- Teams are strongly encouraged to report bugs as soon as they are found; intentional abuse of an unreported bug may be considered as unethical behavior.
- Teams shall not intentionally manipulate the 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.
- Players may not access the implementation instance of the game or modify any variables of the object. In particular, the api and game objects should not be duplicated or modified in any capacity.
- Simulation requests may only be done manually via the website interface, API calls for simulation are not allowed (even if doable).

7 Revision History

Revision	Date	Changes Made	By
2D-0.9	2018-09-08	Preview release for 2D game for the Kick-off. Has multiple TBR/TBA's.	WRF / ASO
2D-0.9.1	2018-09-21	Added details about: SPHERES Pointing direction, Rendezvous Pointing Bonus Points, Cancel Debris Collision Damage too, New API function	WRF/ ASO




SPHERES


COSSPHERES