ZERO ROBOTICS

HS TOURNAMENT 2016

Italian Virtual Competition

Game Manual

Ver. 2.1.0

SPACE-S (Satellite Positioning and Constructing Entities - SPHERES)

TO: Italia SPAZIO-SPHERES Teams

RE: ATTENTION! CALLING TEAMS TO BUILD SATELLITES!

All engineers in Satellite Positioning and Constructing Entities division,

The Italy Governing System urgently needs space engineers to facilitate relocation of humans to Mars.

As you are well aware, Italy is becoming uninhabitable and we have to relocate to Mars permanently. For a smooth transportation to Mars, we must first scout out the best locations for residences and other buildings. Fastest way of identifying these locations is to build surveying satellites to orbit Mars.

The Italian Government already launched the necessary satellite pieces into the space. Engineering teams are asked to construct these pieces into satellites in “ideal zones”, identified based on current solar and meteorite activity. However, the coordinates of ideal zones are unknown until teams place their three Satellite Positioning System devices and get a reading. Once the SPS are placed, teams can assemble their satellite pieces at their zones.

Good luck to all participating space engineers.

On behalf of the Italian Government,

Alvaro Senza-Otero

==========

RE: RE: ATTENTION! CALLING TEAMS TO BUILD SATELLITES!

New information!

We are informed that the space engineers from our rival company SPAZIO-Y are also trying to build satellites to go to Mars. The team which will be able to construct their satellites fastest will be made a contract. But beware, SPAZIO-Y might want some of the satellite pieces that you’ve already collected for themselves. We wish all engineering teams luck in this competition.

May the best team win.

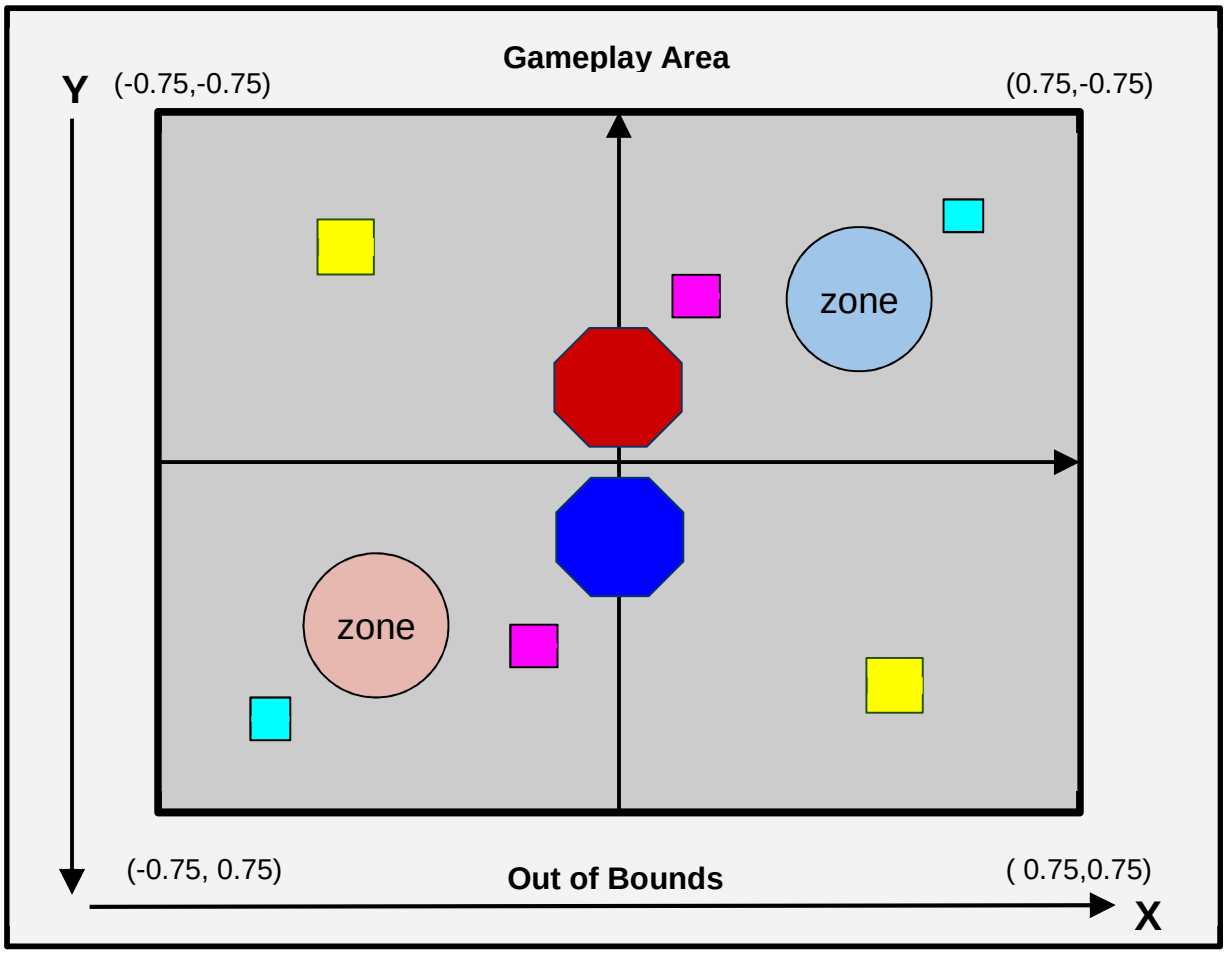
On behalf of the Italian Government,

Alvaro Senza-Otero

ZRHS-Italy 2016 Game Manual

1. Game Overview

Figure 1: Game Overview (with axes corrected and x,y coordinates corrected)



S: Small, M: Medium, L: Large

Diagram not to scale

Matches of SPACE-SPHERES will be played between two SPHERES, controlled by programs written by two separate teams. Each team will compete to have the most points when the round time is up. Each round lasts 180 seconds. Points may be generated by collecting the items (representing the satellite pieces) spread across the playing field and dropping them off inside one's assembly zone, representing the ideal place to construct a satellite.

1.1 Overview of Game Features

This game features Satellite Positioning System (SPS) items, Assembly Zones, and Satellite Piece Items.

1.1.1 Satellite Positioning System (SPS) Items

At the start of the game, there are three Satellite Positioning System (SPS) items given to each SPHERE. Teams need to place these three SPS items far apart to form a triangle; the area of the SPS-triangulated region is used to calculate the error radius that determines the team's estimated assembly zone. The greater the area is, the more precisely they will be informed of the location of their SPACE-S' assembly zone. Read more about SPS items in section 1.4.1 and assembly zones in section 1.1.2.

1.1.2 Assembly Zones

The assembly zone is a spherical location which does not interfere with existing satellite orbits, as chosen randomly from a specific range of coordinates. It is about the size of one SPACE-S, with a diameter of 0.2 meters. Once all three SPS items are placed, the player will be able to use the function `bool game.getZone(float zoneInfo[4])` to learn the estimated zone, that is a spherical zone within a certain radius of error from the true center of their assembly zone. The radius of error of the estimated zone will be calculated as inversely proportional to the area of the triangle formed by the SPS.

Figure 2: SPS and Assembly Zones (with axes corrected and x,y coordinates corrected)

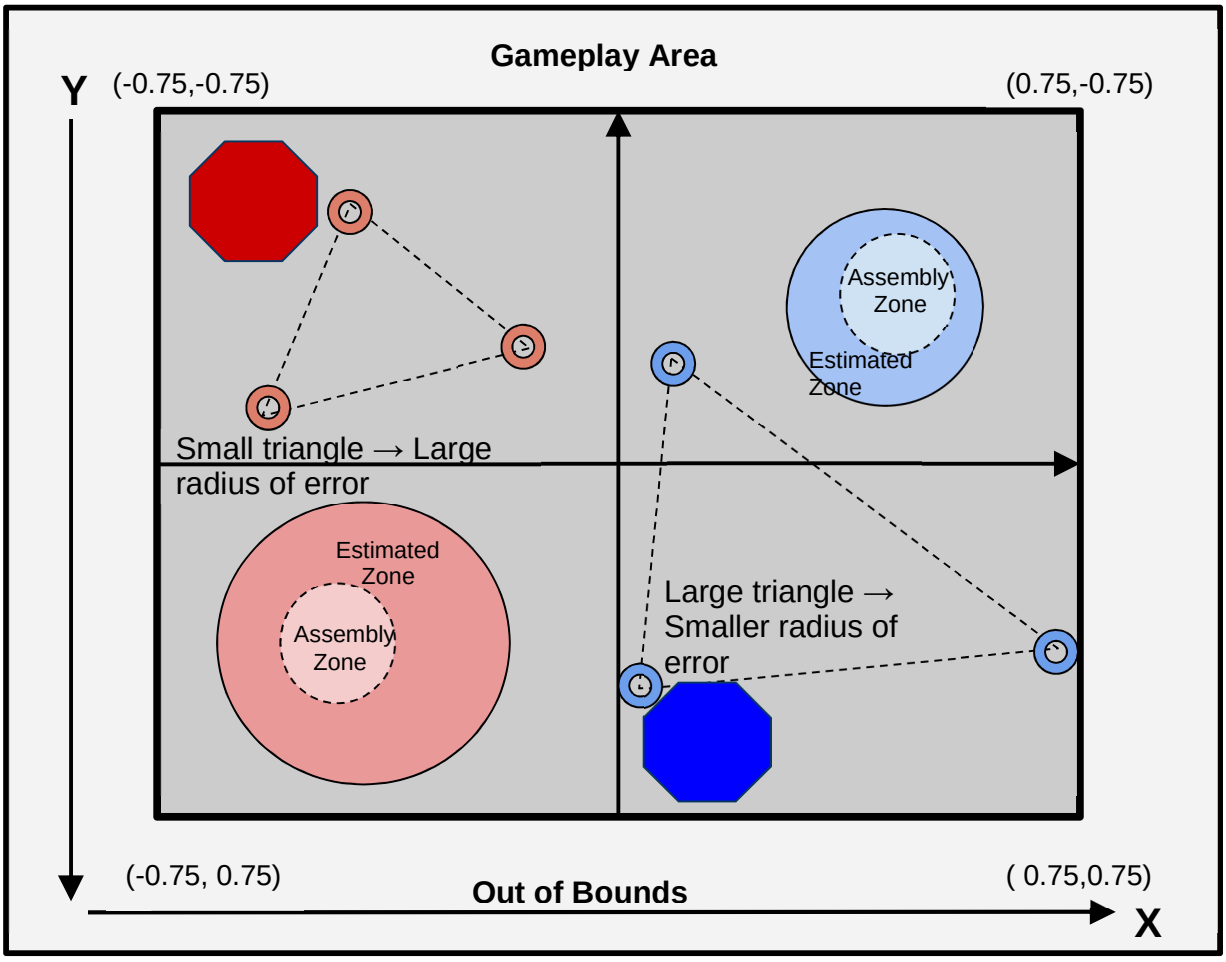


Diagram not to scale

1.1.3 Satellite Piece Items

Once the SPACE-S knows where to assemble the final satellite, it may begin moving satellite piece items to its own specified assembly zone. Satellite items come in three sizes: small, medium, and large. Teams accumulate points for every second that an item is left in their zone. The larger the object is, the more points it will be worth, and the slower you will accelerate while holding it (read in section 1.4.3, 1.4.4). In order for a SPACE-S to pick up an item, it must dock to the item (docking information is in section 1.4.2). The items are distributed symmetrically on the map with the reflection point for each one being about the origin. This way, each item and its reflected pair are the same distance away from each SPACE-S.

1.1.4 Locations of Items and SPACE-S

The locations of satellite items, and the opponent SPACE-S are revealed to the teams. Teams enter the

specified item ID's (described in section 1.4.3) and use the function `void getItemLoc(float pos[], int itemID)` to get the location of the item. A SPACE-S can move anywhere on the map and have fixed initial positions (described in section 1.3.5).

1.1.5 Docking

To pick up a satellite piece, a SPACE-S needs to approach the item, slow down to below .01m/s, be pointing at the item and calling the docking function. The SPACE-S needs to stop in front of the item at a certain distance to pick up and the item will be released at the same distance away (described in section 1.4.2). A penalty is added for docking incorrectly (also described in section 1.4.2)

1.2 Game Layout

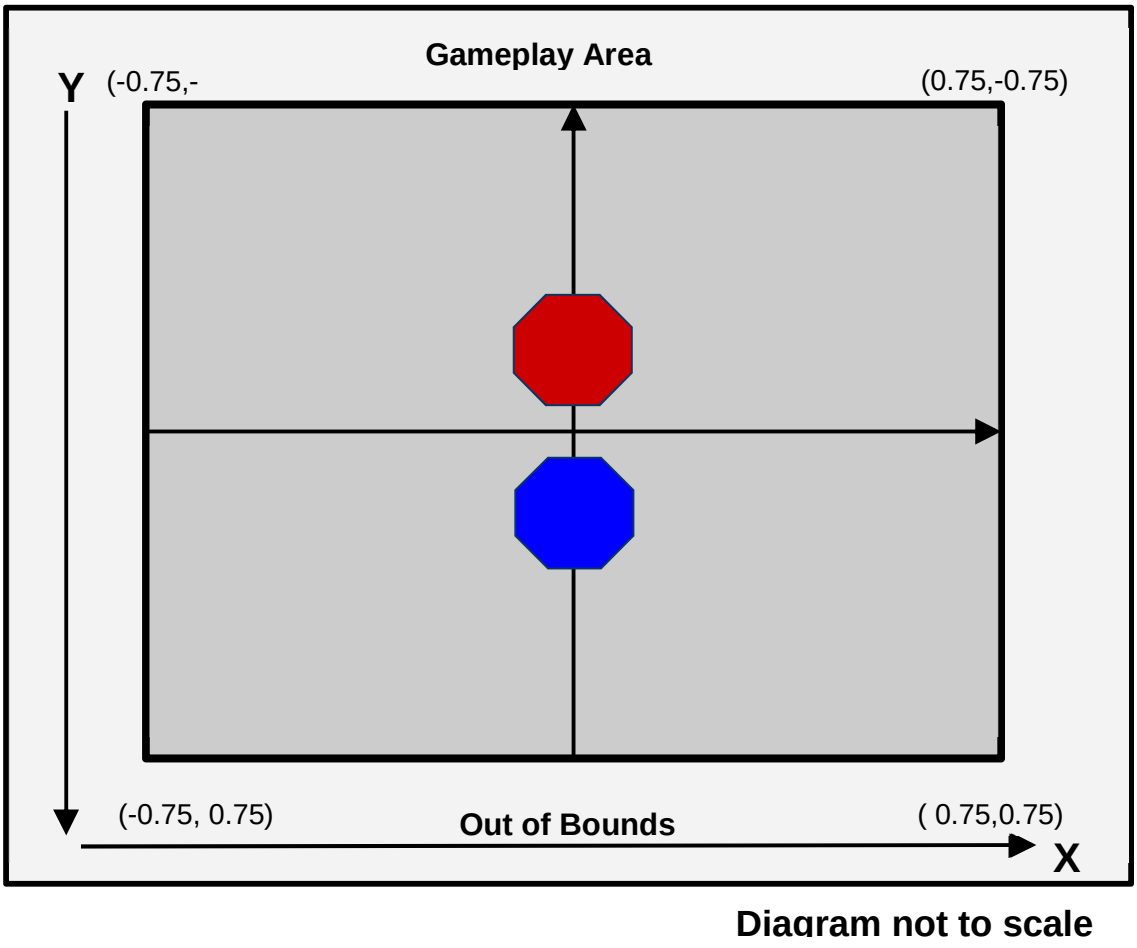
The Zero Robotics High School Tournament Italy 2016 will be conducted in simulation. The game is played in an area called the Interaction Zone. If players leave the Interaction Zone, they will be considered out of bounds. The location of the SPHERES is measured from the center of the satellite.

The Interaction Zone for the game has the following dimensions:

Table 1: Interaction Zone Dimensions

	3D
X [m]	[-0.75 : +0.75]
Y [m]	[-0.75 : +0.75]
Z[m]	[-0.75 : +0.75]

Figure 3: Interaction Zones (with axes corrected and x,y coordinates corrected)



3D Phase

Penalties are added for a) calling the dock function when docking criteria are not met, b) colliding with items, and c) for staying in the opponent's assembly zone for longer than 10 cumulative seconds. See section 1.4.5 for more information about penalties.

1.3 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. The actual SPHERES satellite, like any other spacecraft, has a fuel source (in this case liquid 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 real fuel and batteries by virtual limits within the game. This section describes the limits to which players must adhere to wisely use real SPHERES resources.

1.3.1 ZR User API

Here is a list of functions available in this year's game. Details can be found in the SPACE-SPHERES API (found in Table 2). In order to use these functions, use the syntax game.functionName(inputs).

For example; game.dockItem()

For functions not listed in Table 2, use the syntax api.functionName(inputs), unless they are math functions, which can be called without reference to the instance.

For example; api.setPositionTarget(float posTarget[3])

Generic ZR user API Link: http://static.zerorobotics.mit.edu/docs/tutorials/ZR_user_API.pdf

Table 2: SPACE-SPHERES API Reference

Name	Description
bool dockItem()	Returns true if player picks up an item successfully Attempts to pick up the item closest to the SPHERE.
bool dockItem(int itemID)	Returns true if player picks up specific item successfully Will only attempt to dock with itemID specified @param itemID The integer identifier of a given item.
void dropItem()	Drops the item that the sphere is currently holding
int getNumItem()	Returns the number of total items in play, whether they have been picked up yet or not.
void getItemLoc(float pos[], int itemID)	Copies the location of a given item into the given array. @param pos A pointer to an array of size 3 which will be overwritten by the item location. @param itemID The integer identifier of a given item.
void getItemZRState(float pos[12], int itemID)	Copies the ZR state of a given item into a given array @param pos A pointer to an array of size 12 which will be overwritten by the item state @param itemId is the item whose state is being requested
bool itemInZone(int itemID)	Returns whether the specified item is placed in your zone. Returns false if the item is being held or is not in the zone.
int hasItem(int itemID)	Tells who has a given item. @param itemID The integer identifier of a given item. return 0 if no one has picked up the specified item, 1 if you have picked up item, or 2 if your opponent has picked up the item.
int getItemType(int itemID)	Returns what the item type is. Possible Item Types: * ITEM_TYPE_LARGE * ITEM_TYPE_MEDIUM * ITEM_TYPE_SMALL @param itemID The integer identifier of a given item. return the corresponding item type to the given identifier.
void dropSPS()	Drops SPS Items
float getScore()	Returns player's score
float getOtherScore()	Returns opponent's score

float getFuelRemaining()	Returns remaining fuel
Int getCurrentTime()	Returns the time in the game
void sendMessage(unsigned char inputMsg)	Sends **inputMsg** to other satellite
unsigned char receiveMessage()	Returns the most recent message sent by other satellite
int getNumSPSHeld()	Returns the number of SPSs still held by the sphere
bool getZone(float zoneInfo[4])	If all sps have been placed, returns true and stores a zone estimation in zoneInfo. Otherwise returns false. Can only be called once per second. (In 3D the same values are returned each time the function is called)
bool hasItemBeenPickedUp(int itemID)	Returns true if itemID has been picked up at some point. Returns false otherwise.

1.3.2 Fuel

Each player is assigned a virtual fuel allocation of 60 seconds, which is the total sum of fuel used in seconds of individual thruster firing. 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. Any action that requires firing the thrusters such as rotating, accelerating or moving consumes fuel.

Table 3: Fuel Allocation

	3D
Fuel Allocation [s]	60s

1.3.3 Inter-satellite Communications

The satellites have the ability to communicate with each other using binary messages. The API functions sendMessage and receiveMessage may be used to send data between the satellites. The bandwidth available to the satellites is as follows: (adding function for cooperation)

Table 4: Inter-satellite communications bandwidth

	3D
Message size	Unsigned char

1.3.4 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.

1.3.5 Initial Position

The Blue Sphere starts at the X, Y, Z of [0.0,- 0.15, 0.0]. The Red Sphere starts at the X, Y, Z of [-0.0,- -0.15, 0.0].

The satellite radius is 0.11m, but satellite position relative to game features is determined by the location of the center of the satellite.

Table 5: Initial Positions

	3D
Red	
X [m]	0.0
Y [m]	-0.15
Z[m]	0.0
Blue	
X [m]	0.0
Y [m]	0.15
Z[m]	0.0

1.3.6 Player ID

Users will identify themselves as “playerID = 1” and opponents as “playerID = 2” for all games, whether or not they are the red SPHERES or the blue one respectively.

1.3.7 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 perturbations in different ways, resulting in small or even large variations in score. This is fully intended as part of the challenge and reflects uncertainties in the satellite dynamic and sensing models. The best performing solutions will be those that prove to be robust to these variations and a wide variety of object parameters.

1.4 Gameplay

In order to be victorious over the opposing team, each satellite should place their three SPS items, dock satellite items to place in their zones, all while managing their fuel, their time, and their location on the gameplay area.

1.4.1 Satellite Positioning System

There are 3 SPS items held by each SPACE-S at the start of all the games. The SPACE-S teams must place them to triangulate their zone's center. SPS zone locating works only after all 3 SPSs have been deployed using the void dropSPS() function. SPS calculate a radius of error inversely proportional to the area of the triangular SPS items and displays the zone with this radius of error. Upon dropping your final SPS you will receive points equal to .1/(error radius).

SPS items cannot be placed outside game boundaries.

Table 6: Accelerating Rates with SPS items

No SPS	1 SPS	2 SPS	3 SPS
acc	8/9*acc	⅘ *acc	8/11*acc

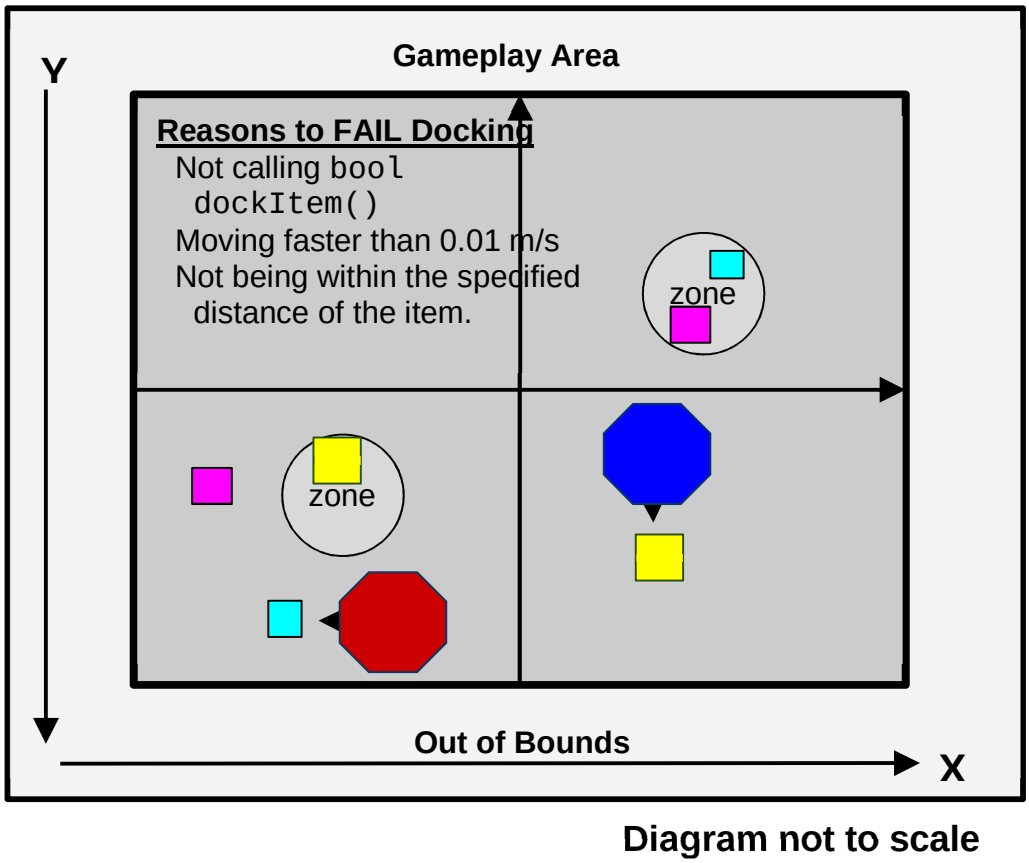
1.4.2 Docking

To dock with an item a SPACE-S must approach an item, not be moving faster than .01m/s, be pointing at the item with a tolerance of 0.25 radians (The angle between the satellite's facing vector and the vector between the center of the satellite and the center of any face of the item is within 0.25 radians), be within a specified distance of an item and call the `bool dockItem()` function. A penalty of .3 points is imposed on a SPACE-S that attempts to dock (calls `dockItem()` function) while not fulfilling the docking requirements. Read more about penalties in section 1.4.5.

Table 7: Item Pickup Distances (meters from center of item)

Item Type	3D
Small	[.124 : .146]
Medium	[.138 : .160]
Large	[.151 : .173]

Figure 4: Docking Satellite Items (with axes corrected and x,y coordinates corrected)



1.4.3 Items

There are six Satellite items (of three sizes) scattered around the interaction zone. Each has a unique numeric identifier from 0 to N-1, where N is the number of items. In the 2D version there are six total items (0-5-).

Table 8: Item IDs and Location

Item Number	Item Type	3D Location(X,Y,Z)
0	Large	(.23, .23, .23)
1	Large	(-.23, -.23, -.23)
2	Medium	(.36, -.36, .36)
3	Medium	(-.36, .36, -.36)
4	Small	(-.50, .50, .50)
5	Small	(.50, -.50, -.50)

The satellite items come with three sizes of which there are two each. The Small item has ⅛ size and mass of SPHERES. If collected in the zone, it gives 0.1pts per second and if a SPACE-S carries it, it can accelerate at 8/9 rate. The Medium item has ¼ size and mass of SPHERES. When in the zone it gives 0.15 pts per second and a SPACE-S with the medium item can accelerate at ⅘ rate. Finally the large item has ⅜ size and mass of SPHERES. It gives 0.2 pts per second and makes a SPACE-S accelerate at 8/11 rate.

Table 9: Size and Mass of Items compared to a SPHERES

	3D
Large Satellite Item	3/8
Medium Satellite Item	1/4
Small Satellite Item	1/8

Table 10: Acceleration Rates for SPACE-S with different Items

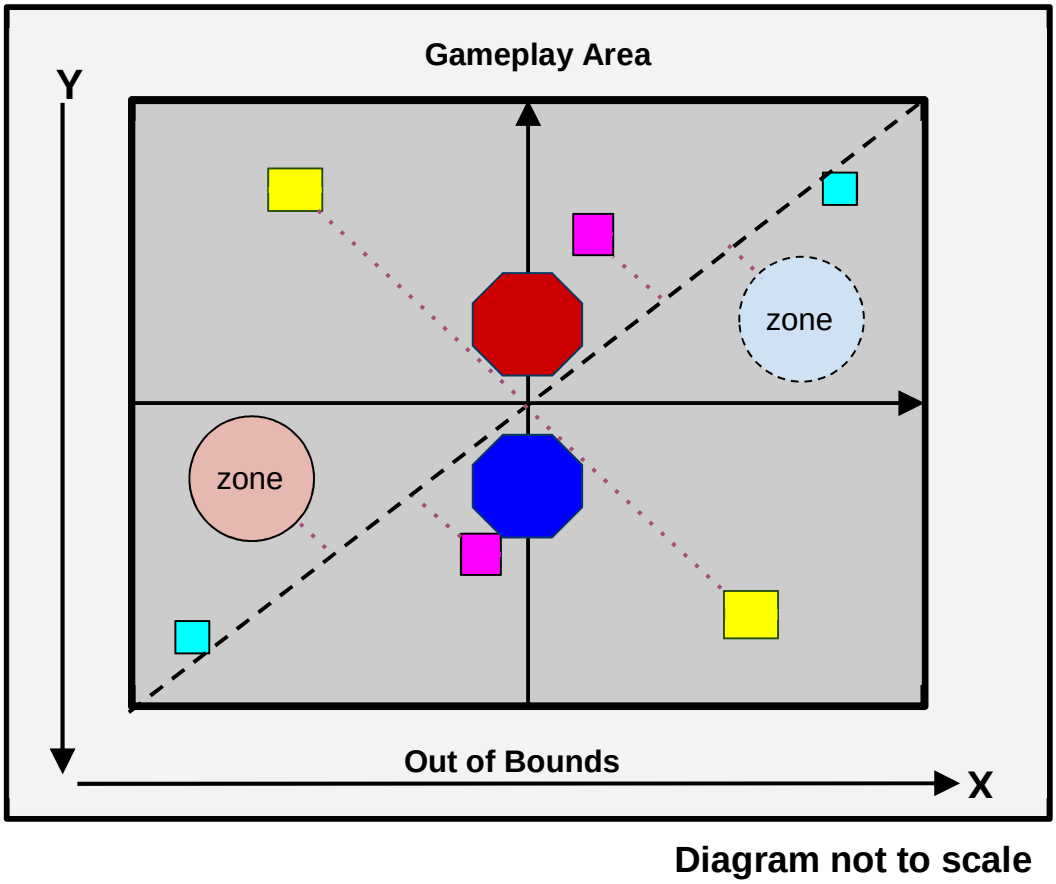
	3D
Large Satellite Item	8/11
Medium Satellite Item	4/5
Small Satellite Item	8/9

Call the game function int hasItem(int item_id) to determine whether the item is held by nobody (0), you (1), or your opponent (2).

A SPACE-S will be penalized if it collides with an item. The penalty will be determined by the SPACE-S's momentum when it collides with the item. This penalty applies everywhere except inside a SPACE-S team's own Assembly Zone where collisions with items will not be penalized. See section 1.4.5 for more information about penalties.

The diagram below shows how the items are placed symmetrically, at the same distance from the diagonal.

Figure 5: Diagonal Symmetry (with axes corrected and x,y coordinates corrected)



1.4.4 Scoring Summary

Your score is largely based on the items you have in your Assembly Zone. Items are worth different point values depending on their sizes.

The scoring calculation is as follows:

Table 11: Point Values

	3D
Small Item [pts/sec]	0.1
Medium Item [pts/sec]	0.15
Large Item [pts/sec]	0.2
Dropping final SPS item	.1/(error radius from SPS)

1.4.5 Penalties

Your score may also change due to various penalties that will be enforced as you play the game.

The penalties will be enforced as follows:

Table 12: Penalties

	3D
Dock incorrectly	-0.3
Collisions with items*	(-5.0 **x** your SPHERES momentum)
Collisions with SPHERES (at velocities below collision avoidance threshold)	(-5.0 **x** your SPHERES momentum. The collision penalty is applied to both SPHERES
Remaining in opponent's zone	-0.4/sec (after cumulative 10 sec)

*There is no penalty applied for collisions with items located inside the player's own Assembly Zone.

1.4.6 End of game

The game ends after 180 seconds. Whichever team has more points wins. In the unlikely case of a tie, whoever is closer to the center/origin of the playing field at game end wins.

2. Tournament

A Zero Robotics tournament consists of several phases called competitions. The following table lists the key deadlines for the 2016 tournament season:

Table 13: Tournament Key Dates

Date (2017)	Event
Until Jan-28th	Registration (by submitting application document)
Jan-14 to Jan-27th	Warmup
Jan-30th to Mar-3rd	Competition
Mar-19th	Final code submission (only the 6 finalists); no leaderboard for this code
End March (TBR)	National Virtual Finals

The rankings in each competition are determined by a Leaderboard, described below.

2.1 The Leaderboard

2.1.1 Introduction

The Zero Robotics Leaderboard has adapted over the years, and is now based on the Whole History Rating (WHR) system. The WHR approach tracks ratings throughout each phase of the competition in order to rank players on the leaderboard. Rating is calculated based on the probability of your team

beating another team during a match (probability of wins and losses). Scores from individual matches are not factored into the calculation. See the “Calculating Ratings” section below for details.

The Leaderboard calculates ratings daily from the beginning of a competition until the submission deadline. All matches during the competition period count toward the rating in the competition. At the end of each competition phase, the final standings on the Leaderboard will determine which teams advance to the next phase.

2.1.2 Playing Matches

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 against other teams submissions, as described in the section titled “Standard play” below. The “Standard Play” cycle is repeated a total of five (5) times using the team placement from the previous cycle as the starting point for the next cycle. Results are posted on the website after all five (5) cycles have been completed. 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 “Standard play” cycles are completed for the following reasons: 1) To settle or converge results daily. (This ensures a more accurate reflection of rank for teams entering a competition leaderboard for the first time.) 2) To provide more match data for teams at the top and bottom of the leaderboard since these teams play fewer matches overall.

2.1.3 Standard play

As a default you will play the 9 teams above you and the 9 teams below you. If there are less than 9 teams above you, you will play all teams above you plus the 9 below you. If there are less than 9 teams below you, you will play all the teams below you and the 9 teams above you . Teams are randomly assigned as Blue or Red SPHERE during each match.

2.1.4 Initial submission rule

When your team first submits code to the competition you will play matches against the bottom quarter of the teams already on the leaderboard. On the first day with submissions, those teams will play as per “Standard Play”.

2.1.5 Last day of the competition

On the last day of the competition, the leaderboard will be run multiple times to converge/ stabilize the results: to calculate entry position of teams submitting for the first time and to assess final placement of all teams.

2.1.6 Calculating Ratings

A team’s standing in a competition is determined by a value called rating. The Leaderboard tracks all matches a team has played against other players in the course of the competition and creates a rating

based on the outcomes.

Your rating is the factor R0 in the equation for the probability of your team beating another payer of rating R1 in a single match.

$$\text{Probability}(W) = e^{R0} / (e^{R0} + e^{R1})$$

The variable R0 is computed using an algorithm based on Bayesian inference. The probability of a rank r given a series of game outcomes G is then:

$$P(r|G) = P(G|r)P(r) / P(G)$$

is the prior distribution of ratings, which is assumed to vary in a random walk process with a standard deviation over each day of the competition of 0.1, chosen after testing for stability of the board. This allows all previous ratings to change, as implied by the name Whole History Rating. Ignoring P(G) because it is a normalizing constant with no effect of the final rating r, and incorporating the standard deviation of ratings, having each day of the competition representing k:

$$\prod P(r|G) = P(g_k|r_k)P(r_k|r_{k-1})$$

The final term P(r 0) is set to 0.

The algorithm uses newton's method to maximize this probability as a function of your rating. The rating at this maximum is your new rating.

2.1.7 Summary

Not all wins are equal. Wins and losses are valued by their relation to the projected win probability. Your rating corresponds to a 50% win probability, meaning you are more than %50 likely to win against teams with lower ratings than yours, but not against teams with higher scores. Unexpected match results, either favorable or unfavorable, will leave the most noticeable impacts on your rating.

Although this process is based on probability, it is wholly reliable. The rank computation algorithm corrects errors in rank history throughout the competition, and optimization and stabilization safeguards have been added to calculate rank with extreme precision. Also, the leaderboard is designed to allow data to propagate throughout the system. If two players don't run against each other because their rank difference is more than 10 slots, overlapping matches update both teams' ratings. Every team's rating is relative to all other ratings.

To summarize -- there are several factors that affect a team's rating:

· Match Outcomes: A team that consistently wins matches will usually have a higher rating.

· Opponent Rank Winning against a higher ranked team will usually improve a team's rating.

· Other Match Outcomes The Leaderboard takes into account all matches played by all teams. Even if two teams do not have a direct encounter, their match outcomes will have an effect as they filter through third parties.

The team with the highest rating will be rank 1 on the leaderboard. The best way to stabilize and even 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 match ruining your team's prospects keep you from submitting early. Even though every submission is factored into your match history, results of past competitions

demonstrate that teams that make many submissions tend to perform better.

Finally, it is important to note that after every competition phase, (2D, 3D, Alliance phase) all ranking data is refreshed and teams start from scratch.

2.2 2D Warmup

All teams that complete a valid registration are eligible to participate in the warmup phase.

2.3 3D Simulation Competition

All teams that complete a valid registration and submit code that achieves a non-zero score when competed against a "blank player" (a player with no code) by the warmup deadline are eligible to participate in the 3D simulation competition.

2.7 Virtual Final Competition

The top 6 teams on the leaderboard at the end of competition will advance to the Virtual Finals Competition.

The virtual finals will take in simulation during the final event (TBD). Teams will be invited to participate to the event.

2.7.2 Competition Format

The teams will be divided into 2 conferences for the virtual competition. All teams ranked with odd numbers will participate in Conference A; all teams ranked with even numbers will participate in Conference B, as shown in this figure.

Figure 8: Division of Teams between Conferences

Conferenze A - ranks	Conference B - ranks
1,3,5	2,4,6

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 the Virtual Competition Bracket figure.) The following rules determine the winner:

1. The team with the most wins advances

2. If alliances are tied for wins, the alliance with the highest total score advances

3. If scores are tied, one more simulation will be run to break the tie.

The two winners for each bracket will compete against each other for 1st and 2nd positions. If scores differ less than a threshold (TBD), a second match will be run. If both agree, winner is clear; if they disagree and the second differs more than the threshold, the winner of the second run is the champion; if both are below the threshold, a third match is run; the winner of this is the champion, whatever is the

score.

The second-ranked for each bracket will compete against each other for 3rd and 4th positions. Same rules apply as for the 1st vs. 2nd match.

3. Season Rules

3.1 Tournament Rules

All participants in the Zero Robotics High School Tournament 2016 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.

3.2 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).

4. Revision History

Revision Number	Date	Changes Made
2.1.0	14 Jan. 2017	Adapted to Italian Virtual Championshio