ZERO ROBOTICS

HS TOURNAMENT 2015

Game Manual

Ver. 3.1.0

FWD: FWD: FWD: URGENT! ALL EMPLOYEES NEEDED!

Hello employees of BLU Industries, I have exciting news for all of you. One of NASA’s old satellites has malfunctioned and spontaneously broken into pieces in low Earth orbit. It was holding very valuable data. NASA has stated that any company that wishes to try their luck at gathering this data is free to do so. This represents a huge opportunity for BLU!

In order to be the first company out there, we’ve outfitted our HYPER-SPHERE™ with the newest propulsion and control systems from R&D, products of trillion-dollar research. It is powered by solar energy, so it can recharge whenever it’s under the sun.

Members of the BLU family, I’m calling on you today to develop an autonomous piloting system for the HYPER-SPHERE™, so that when it reaches the NASA satellite it will be able to collect as much data as possible as quickly as possible. While doing so, your pilot should also be wary of how much energy it has, recharging from the sun or from the strewn about batteries of the broken satellite as needed.

Thank you for your work,
Alvar Saenz-Otero
BLU CEO

==
===

RE: FWD: FWD: FWD: URGENT! ALL EMPLOYEES NEEDED!

Hello again employees, the game has changed. I have just received news that RED corporation, led by their CEO Evil Alvar, have made their own plans to recover the orbiting data. Were it any other company, I would not be worried, as our HYPER-SPHERE™ would easily get there first. Unfortunately, however, RED have their own MEGA-SPHERE™, and I have a hunch that they too will be pulling out their newest technology for this venture. It looks as though we’ll have some competition.

There is, however, a silver lining to this new development. If our R&D department can get their hands on information about RED’s new transportation technology, especially pictures, it would be even more valuable to us than NASA’s data. Thankfully, the HYPER-SPHERE™ is already equipped with a camera, although it isn’t powerful enough to take pictures without light from the sun.

Therefore, your job has expanded. While collecting the debris is still important, your pilot should focus on gaining intel about our competitor’s satellite. We also fear that RED will attempt to take pictures of us, so additionally work to prevent that as best you can.

Let’s get that tech!
Alvar Saenz-Otero
BLU CEO

Table of Contents

ZRHS 2015 Game Manual

1. Game Overview

Figure: Game Overview

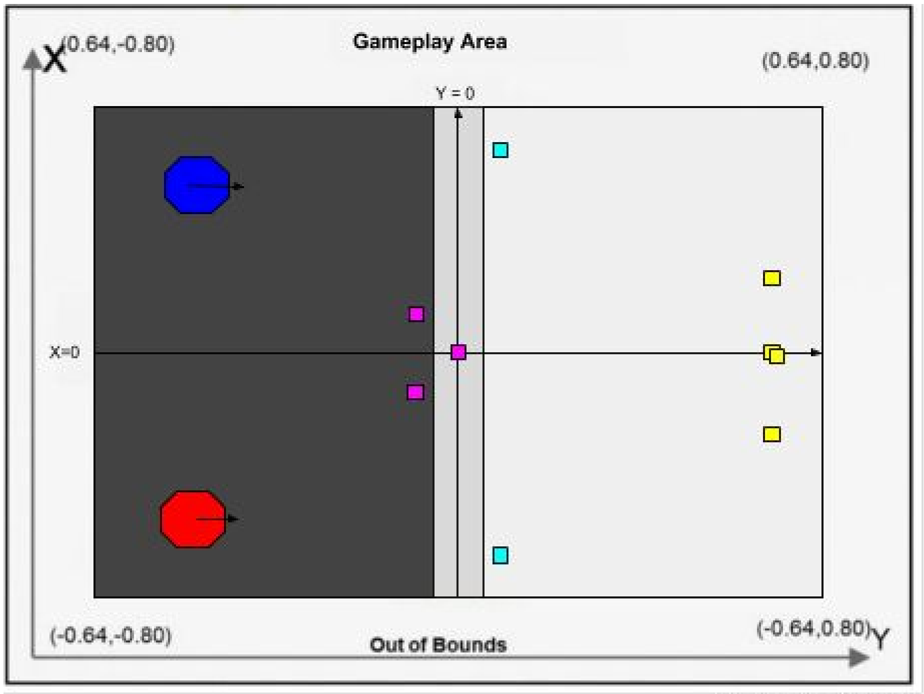


Diagram not to scale

Matches of SPY SPHERES will be played between two SPHERES satellites, controlled by programs written by two separate teams. Each team will compete to have the most points when the round time is over. Each round lasts 180 seconds. Points may be generated by taking pictures of the other satellite and uploading them, or by collecting one of the score-generating items (representing the pieces of the NASA satellite) spread across the playing field.

In this game, there are dynamic Light and Dark zones that have various impacts on the satellites. These represent how a satellite in space, in Low Earth Orbit, is half of the time illuminated by the sun and the other half in Earth's shadow (also called eclipse).

Additionally, real space satellites have small amounts of power available for all of their equipment. Therefore, in SpySPHERES, each satellite has a finite amount of energy for any game actions. Real satellites launch with batteries and use solar panels to replenish their energy when exposed to direct sunlight. In SpySPHERES, energy can be generated by being in the Light Zone or by picking up energy items.

Pictures cost energy to take and upload, and can only be successfully taken when the target is in the Light Zone. The satellites have a storage capability of 2 pictures, which must be uploaded to receive the corresponding points. In order to upload the pictures in 3D, the satellites must be facing Earth, which is in the positive Z direction.

The Light, Dark, and Grey zones will change positions over the course of the round.

Lastly, there are mirror items which, when deployed, prevent the user from taking pictures but also reflect any pictures your opponent takes back at them, making them worthless.

1.1 Game Layout

The Zero Robotics High School Tournament 2015 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:
X: [-0.64: +0.64]
Y: [-0.8: +0.8]
Z: [-0.64: +0.64]
Satellites that go out of these limits will lose 2% of their initial fuel per second.

Figure: Interaction Zones

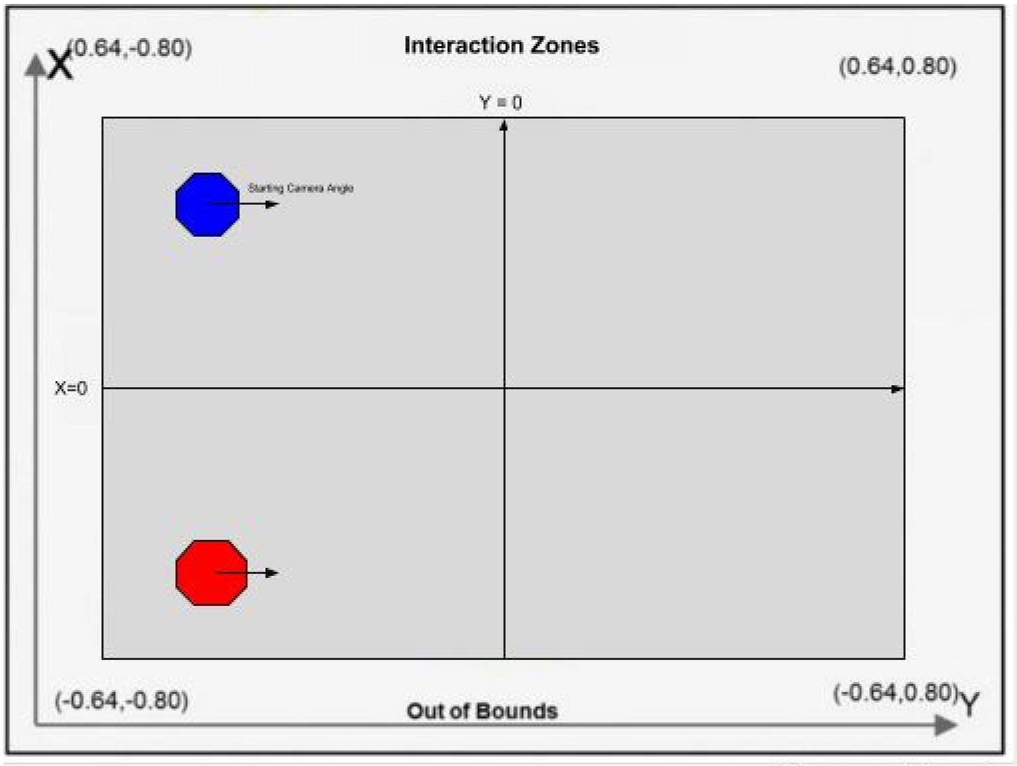


Diagram not to scale

2D Phase:

For the 2D game, the arena is a plane with only X and Y cardinal dimensions. The light and dark each take up either the negative or the positive half of the Y-axis, and switch positions. The grey is between them. There is more info on the light and dark zones in all phases in 1.3.2. There are also nine items, seven Score+ and Energy Pack types and two Mirror types, information on which is in 1.3.3.

3D Phase:

From the 3D phase onwards, the Z dimension is opened up for movement. In order to upload the pictures taken, the satellite must face towards Earth, in the positive Z direction. With more degrees of freedom, there are more strategies that you can try your hand at. Good luck!

Alliance Phase:

For the alliance phase, the grey zone is split into two, as the light and dark zones now move and cycle around the interaction zone, as further explained in section 1.3.2 of this document. Furthermore, item locations are randomized within parameters detailed in section 1.3.3. Finally, the Alliance phase features harsher penalties for going out of bounds in the form of grey-zone like energy costs.

Mechanic Summary Table

	2D	3D	Alliance
Score Items	3	4	4
Mirror Items	Pictures are worth 0	Pictures are worth Negative Points	Pictures are worth Negative Points
Energy Items	4	3	3
Item Locations	Deterministic	Deterministic	Randomized
Light Zone	Switching	Switching	Moving Continuously
Uploading	No Attitude Req.	Must be facing Positive Z	Must be facing Positive Z
Out of Bounds	Light is the Same as In Bounds	Light is the Same as In Bounds	Light is the Same as Grey Zone.

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 it’s 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.

1.2.1 ZR User API

Here is a list of functions available in this year's game. Details can be found in the API.
In order to use these functions, use the syntax `game.functionName(inputs)`. For functions not listed here, use the syntax `api.functionName(inputs)`, unless they are math functions, which can be called without reference to the instance.

```
float getFuelRemaining();
int getMemoryFilled();
bool posInLight(float pos[]);
bool posInDark(float pos[]);
bool posInGrey(float pos[]);
int posInArea(float pos[]);
int hasItem(int itemID);
float takePic();
float uploadPics();
float getPicPoints();
int getMemorySize();
bool isCameraOn();
float getScore();
float getOtherScore();
int getLightSwitchTime();*
void sendMessage(unsigned char inputMsg);
float getDarkGreyBoundary();
float getLightGreyBoundary();
float getLightInterfacePosition();
unsigned char receiveMessage();
void getItemLoc(float pos[], int itemID);
int getCurrentTime();
bool isFacingOther();
float getEnergy();
float getOtherEnergy();
int getItemType(int itemID);
int getNumItem();
int getNumMirrorsHeld();
bool useMirror();
```

* No longer relevant in the Alliance version of the game. Deprecated.

API Link: http://zerorobotics.mit.edu/tournaments/20/info/100/0/

1.2.2 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 or moving consumes fuel. Being out of bounds consumes an additional penalty of 2% of initial fuel per second.

1.2.3 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 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.

1.2.4 Initial Position

The Blue Sphere starts at the X, Y, Z of [0.4,-0.6, 0.0].
The Red Sphere starts at the X, Y, Z of [-0.4,-0.6, 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.

1.2.5 Player ID

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.

1.2.6 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.

1.3 Gameplay

In order to be victorious over the opposing team, each satellite should make use of the consumable items and their camera in order to take pictures of the opponent and gain points, all while managing energy, fuel, memory, and their position in light or darkness.

1.3.1 Energy

Energy is the most prohibitive resource the satellites utilize. Both players start with 5.0 energy at the start of the game, which is also the maximum energy a satellite can have.. If the satellite is in the light zone, it gains 0.5 energy every second. To maneuver the satellite, 0.15 energy is used per 1 second of fuel used. Other activities that cost energy include taking and uploading pictures.

The satellites can check how much energy it has left by calling the game function
`float getEnergy()`

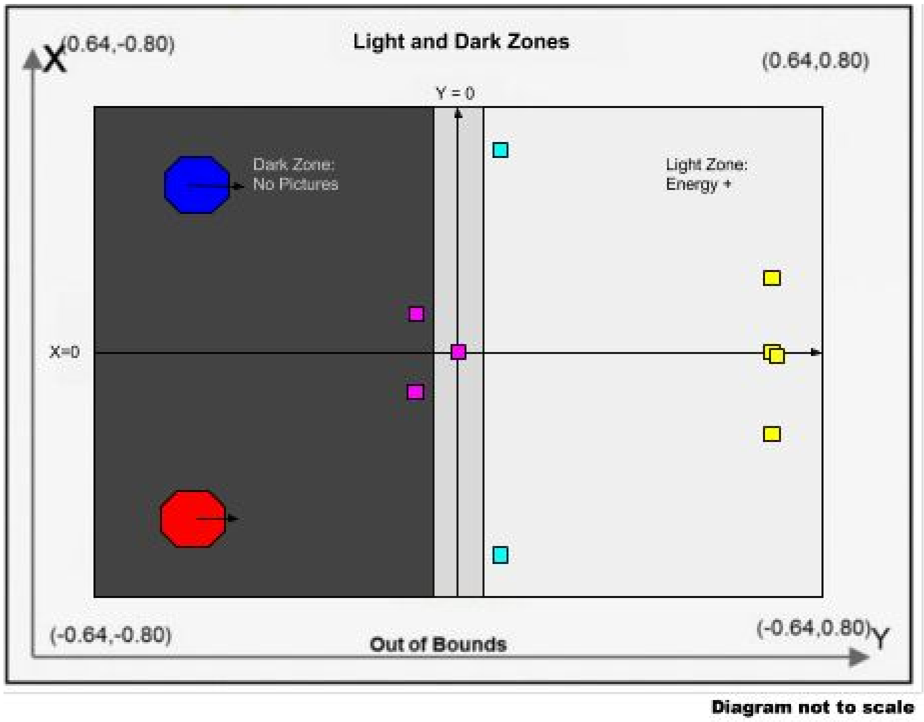
The satellite may also check the energy of the opponent by calling `float getOtherEnergy().`

1.3.2 Light and Dark Zones

The Light and Dark Zones, and the Grey Zone in between them, will be the main determinant of how well you can take pictures of the other satellite. Here is a table with their properties:

	Picture Can be Taken of Me	Does Energy Recharge
Light Zone	Yes	Yes
Grey Zone	Yes	No
Dark Zone	No	No

Figure: Light & Dark Zones



2D and 3D Version:

The negative Y sector starts out in the Dark Zone and the positive section starts out in the Light Zone. In between them, with a total width of 0.2 meters centered at the origin, is the Grey Zone. Between 30 and 60 seconds into the game, the Light Zone and the Dark Zone will switch position. The zones will switch again periodically every 60 seconds until the end of the game.

The user can call the function `int getLightSwitchTime()` to determine how long, in seconds, until the next zone switch. This mechanic works exactly the same as it did in the 2D version of the game.

Alliance Version:

In the Alliance Version the light becomes dynamic. Similarly to the other modes, there is a Light Zone and a Dark Zone, with Grey Zones where the two meet. Now, however, there are two Grey Zones, each of which is half as wide as before, at 0.1 meters. All of these zones are constantly moving along the Y-axis at 0.025 meters per second. The negative and positive ends of the y-axis act as though they are connected, so when the light would go out of bounds in +Y, it instead wraps around and comes out of -Y.

The border between the Light and Dark Zones where light is in the positive Y direction serves as the reference point. It is easy to think of it as the tail end of the Light Zone. This point begins at y=-.2, and all other aspects of the light are constant relative to it. The game function `float getLightInterfacePosition()` returns the current location of this light boundary.
All out of bounds areas are treated as the Grey Zone, in addition to draining fuel.

1.3.3 Items:

There are three types of items scattered around the interaction zone: Energy Packs, Score+, and Mirrors. Each has a unique numeric identifier from 0 to N-1, where N is the number of items.

There are nine total items (0-8), with all three item types. Each of them may only be used once, and they do not regenerate.

Item Types:

- Energy Pack - Upon pickup this item is instantly used. It refills the satellite to its maximum energy level, 5.0.
- Score+ - Once picked up this item is also used immediately. It adds 1.5 to the satellite's score.
- Mirror - Unlike the other two items, this is simply held on to once picked up. Once it is deployed, the holder has 24 seconds during which they cannot take pictures, but any pictures the opposing satellite takes of them will be worth no points in 2D, and in 3D a negative value whose magnitude is equal to the number of points the picture would have otherwise been worth. To clarify, the mirrors do not prevent pictures from being uploaded, merely from being taken. Some related functions are:
  - `void useMirror()` - Deploys a mirror.
  - `int getNumMirrorsHeld()` - Returns the number of mirrors the user has.

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

Call the game function `float[3] getItemLoc(int item_id)` to obtain the location of the item.
Several other item-related functions are detailed in the API.

In order to collect an item, the center of the user's satellite must be no greater than 0.05 meters away from the item and it must be moving at slower than .01 meters per second.

Figure: Item Collection

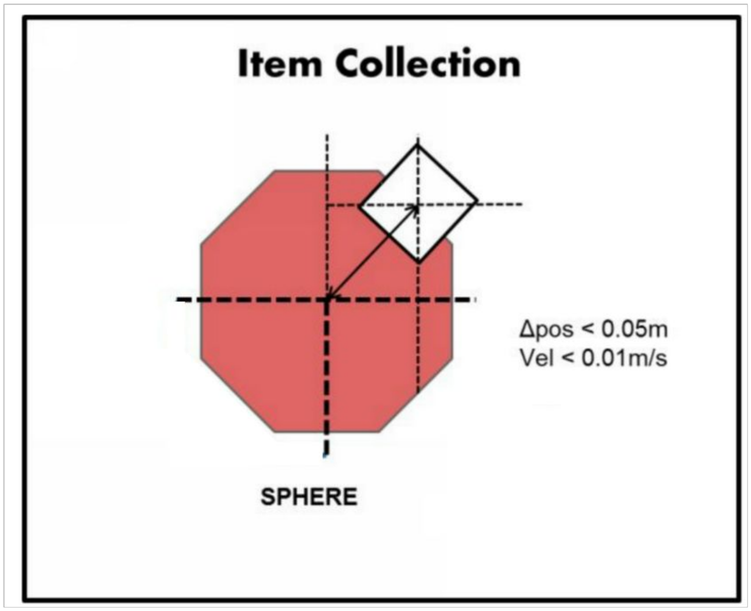


Figure: 3D Item Locations & Types

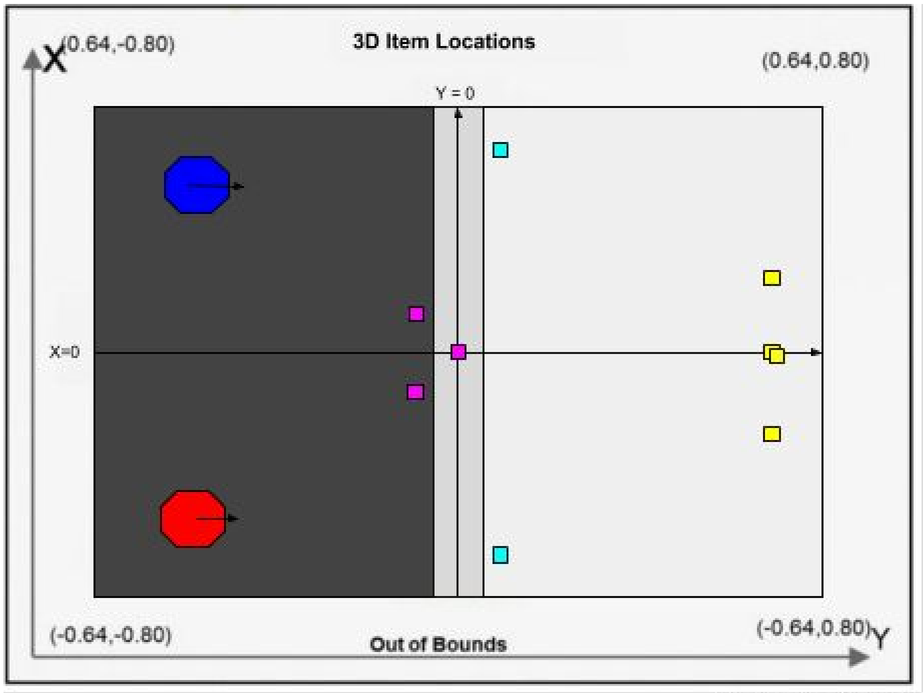


Diagram not to scale

Table of Item Locations (2D/3D):

Item Locations			
ID	**Item**	**2D Location**	**3D Location**
0	Energy Pack	(0.3, 0.0)	(0.3, -0.2, 0.3)
1	Energy Pack	(-0.3, 0.0)	(-0.3, -0.2, 0.3)
2	Energy Pack	(0.0, 0.3)	(0.0, 0.0, 0.3)
3	Energy Pack	(0.0,-0.3)	N/A
3	Score+	N/A	(0.0, 0.6, 0.4)
4	Score+	(0.0, 0.6)	(0.4, 0.6, 0.0)
5	Score+	(0.4, 0.6)	(-0.4, 0.6, 0.0)
6	Score+	(-0.4, 0.6)	(0.0, 0.6, -0.4)
7	Mirror	(0.6,-0.7)	(-0.4, 0.15, -0.4)
8	Mirror	(-0.6, -0.7)	(0.4, 0.15, -0.4)

Alliance Items:

In the Alliance Phase, items locations will be randomized within certain bounds, as seen on the chart below. In order to keep the game fair, all asymmetries are mirrored across the Y axis. Use `void getItemLoc(float pos[], int itemID)`, in order to find location.

Alliance Item Location Ranges

ID	Item	X	Y	Z
0	Energy Pack	0.175 ± 0.125	0.15 ± 0.15	0.25 ± 0.25
1	Energy Pack	Position of Item 0, Mirrored Across the Y Axis		
2	Energy Pack	0.0 ± 0.0	0.15 ± 0.15	0.25 ± 0.25
3	Score+	0.175 ± 0.125	0.50 ± 0.10	0.0 ± 0.50
4	Score+	Position of Item 3, Mirrored Across the Y Axis		
5	Score+	0.175 ± 0.125	0.50 ± 0.10	0.0 ± 0.50
6	Score+	Position of Item 5, Mirrored Across the Y Axis		
7	Mirror	0.30 ± 0.10	0.15 ± 0.15	-0.40 ± 0.10
8	Mirror	Position of Item 7, Mirrored Across the Y Axis		

1.3.4 Picture Taking

The way to score the largest possible amount of points is by taking pictures of the other satellite. There are multiple requirements for taking a picture:

- The user satellite must be facing the opposing satellite. (The angle between the satellite's facing vector and the vector between the positions of the two satellites is within 0.25 radians.)
- The opposing satellite must not be in the Dark Zone (that is, it must be in the Grey Zone or the Light Zone).
- The user satellite must have at minimum 1 energy.
- The user satellite must have at least 1 open memory slot. Each satellite has two memory slots total.
- The user satellite's camera must be on.
- The user must not have an **active** mirror item.
- The user's satellite must be at least 0.5m away from the opposing satellite.
- The user must call the function `float takePic()`.

The function `float takePic()` will then take a picture if all of the conditions are met, and store it in the first available memory slot. Regardless of whether the picture was successfully taken, `takePic()` will consume 1 energy and shut down the camera for 3 seconds when it is called. The function will return 0 if picture-taking was unsuccessful, and the point value of the picture otherwise. Unsuccessful pictures do not use a memory slot. Every picture, whether successful or not, adds 0.01 points to the user's score as a tiebreaker.

The amount of points each picture is worth is determined by the distance between the two satellites when the picture is taken. It follows this formula:

```
points = 2.0  + 0.1 / (distance - PHOTO_MIN_DISTANCE + 0.1)
```

Where `PHOTO_MIN_DISTANCE` is 0.5. If the opposing satellite is using an active mirror, the amount of points the picture is worth will zero in 2D, and the opposite of the above value in 3D and Alliance.

The function `float getPicPoints()` is available to check the points a picture will be worth before taking it. Calling it costs 0.1 energy, but does not disable the camera. Note that this function can sense whether or not the opponent has deployed a mirror.

Figure: Reasons to Fail a Picture

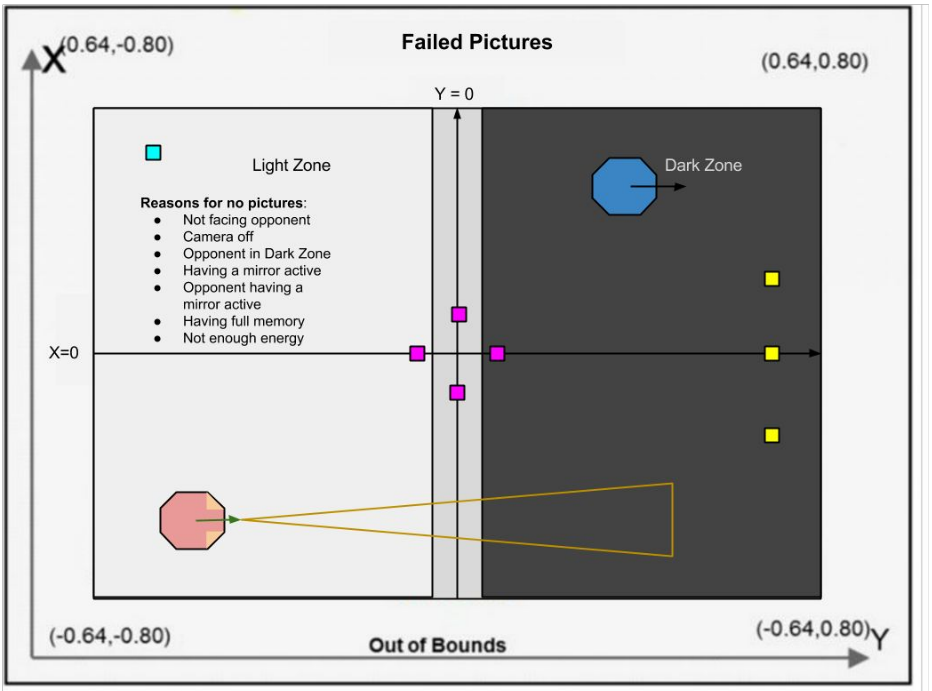


Diagram not to scale

Figure: Conditions for a Successful Picture

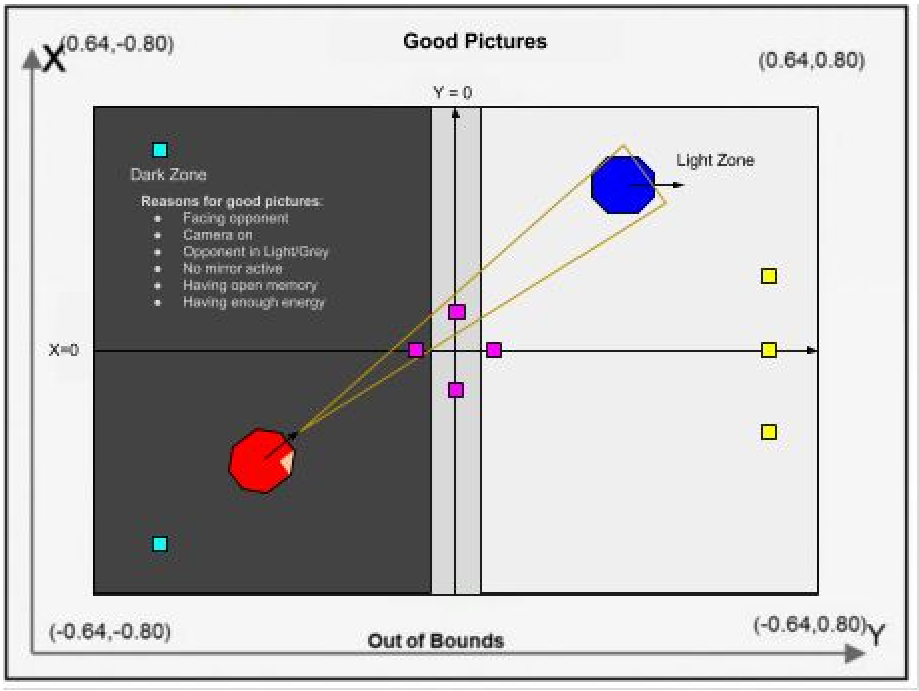


Diagram not to scale

1.3.5 Uploading Pictures

In order to receive points for the pictures that have been taken, the sphere needs to upload them to Earth. When uploading there are also several requirements that must be met:

- The satellite must have at least 1.0 energy.
- The satellite must call `float uploadPics()`.

When calling the function, the camera will be disabled for 3 seconds, similarly to taking pictures Afterwards, `uploadPics()` will return the number of points that were uploaded. This process costs 1.0 energy. You do not receive points for pictures taken that are not uploaded.

In addition, for the 3D and Alliance games, you must face towards Earth. The conditions for facing Earth are as follows:, you must be facing towards the positive Z direction to within a tolerance of 0.25 radians, and you must not have a faster rate of change of the attitude vector than 0.05 rads/sec.

1.3.6 Scoring Summary

Method	Point Value
Attempting to Take a Picture (Valid or Not)	0.01
Taking a Valid Picture	0.1
Uploading Pictures	$2.0 + \frac{0.1}{distance - 0.5 + 0.1}$ per picture without a mirror, and in 3D and Alliance the opposite of that per picture with a mirror.
Score Items	1.5 per item

Whoever is closer to the center/origin of the playing field at game end wins in the event of a tie.

1.3.7 End of game

The game ends after 180 seconds. Whichever team has more points wins. In the unlikely case of a tie, the satellite that is closer to the origin wins.

2. Tournament

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

Table of Tournament Key Dates

Sep 12 (Sat)	Kick-off webcast
Oct 2 (Fri)	2D Simulation Competition Deadline (end of practice)
Oct 30 (Fri)	3D Simulation Competition Deadline
Nov 7 (Sat)	Alliance Formation Event
Dec 4 (Fri)	Semifinal Simulation Competition Deadline

Dec 20 (Sun)	ISS Code Due
TBD	ISS Final Competition

The rankings in each competition are determined by a *Leaderboard*, described below. The 2D Simulation Competition is *not* an elimination round; almost everyone who submits code which achieves a nonzero score against a “blank” player (player with no code) to the 2D Simulation Competition advances to the 3D Competition.

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 16:59:59 ET/ 20:59:59 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:

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

P(r) 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 Practice Simulation Competition

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

2.3 3D Simulation Competition

All USA, Russian, and pilot 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 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.

2.4 Alliance Formation Event

The top ranked 83 teams on the leaderboard at the end of 3D simulation play plus 1 wild card (a total of 84 teams) 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 83 will be invited to participate in the Virtual Finals. See Section 2.8 for details about the Virtual Finals.

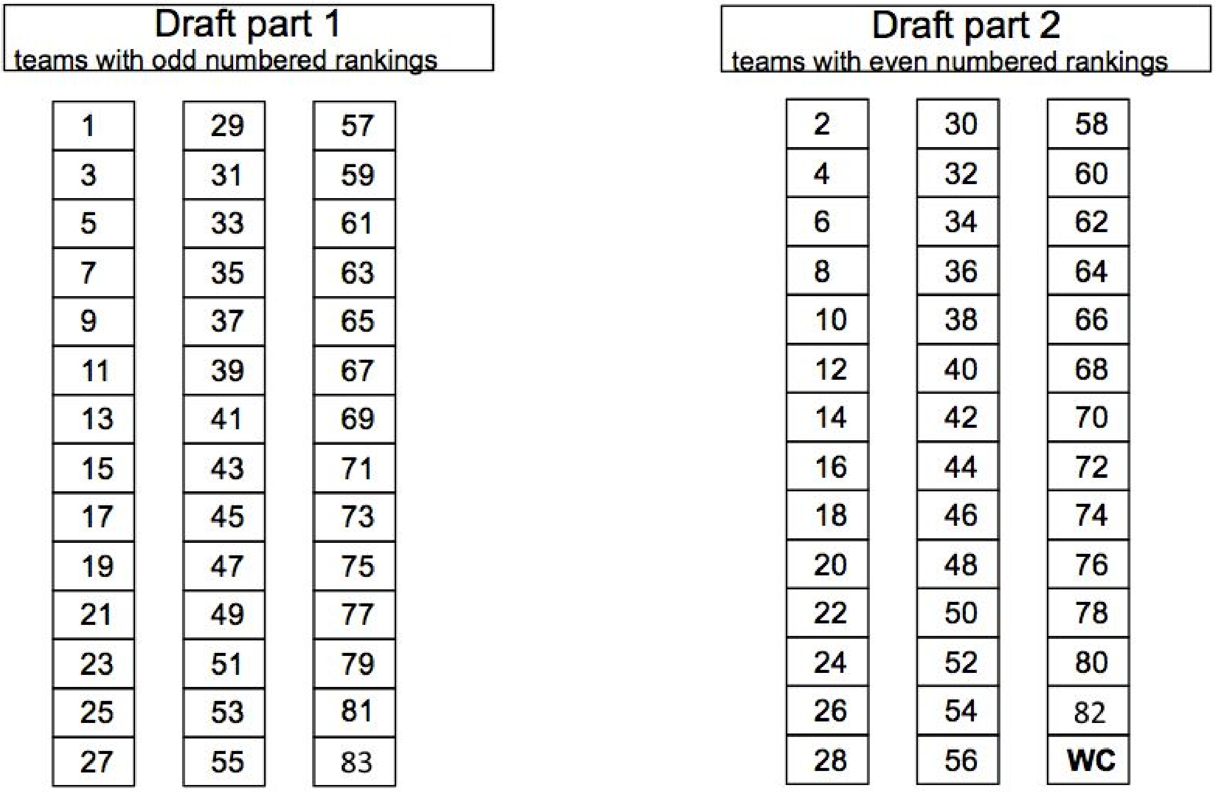
The ranking of the teams will be determined by the rankings at the close of the 3D leaderboard.

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 7th. 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.

Figure: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	**WC**

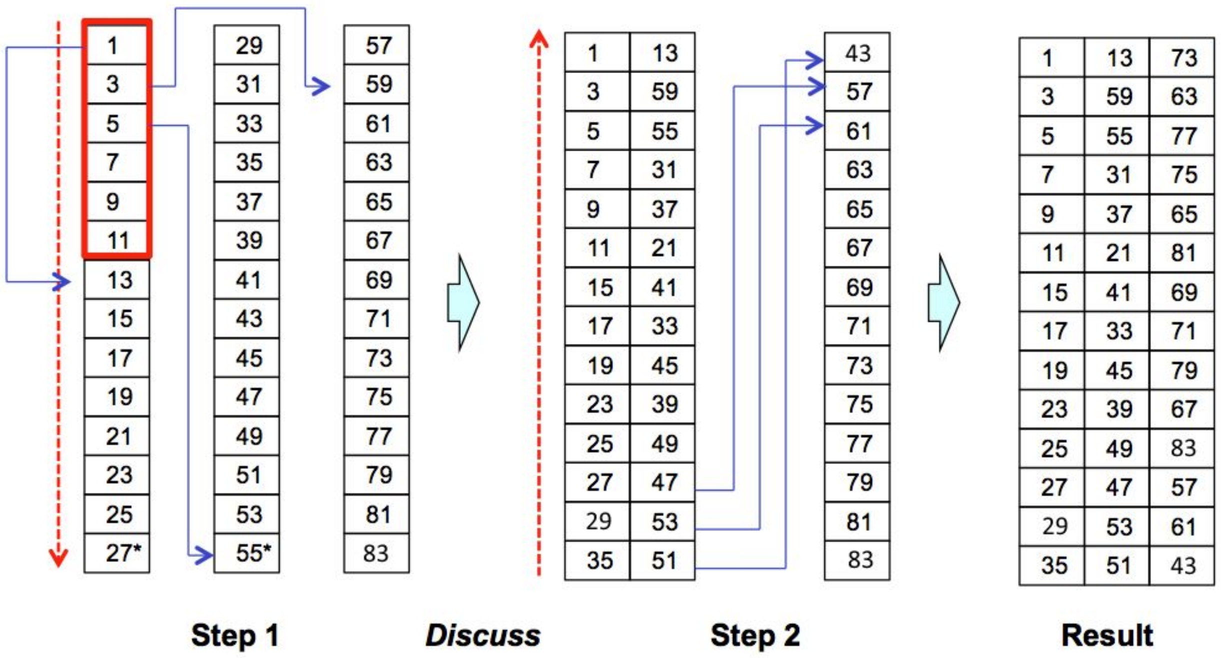
Note: The times of the draft parts are To Be Announced

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.

Figure: Alliance Creation Process- demonstrated for teams with odd number rankings



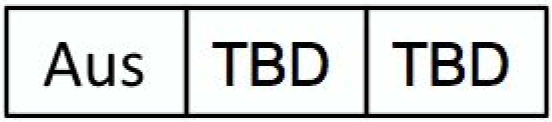
1	13	73
3	59	63
5	55	77
7	31	75
9	37	65
11	21	81
15	41	69
17	33	71
19	45	79
23	39	67
25	49	83
27	47	57
29	53	61
35	51	43

(*The example shown above is approximate.)

2.5 Wild Cards Explained

This year we welcome our Australian partners to the Zero Robotics High School Tournament 2015. As this is their pilot year, at least one Australian alliance (composed of one Australian team and two other teams, TBD) will participate in the ISS competition as shown in the figure below. Therefore we have reserved a spot for the Australian team out of the total 84 teams advancing to the alliance phase. Since we cannot predict the ranking of this team during the 3D competition, 1 wildcard spot has been included. The pilot team will advance whether or not they place within the top 83 ranked teams on the leaderboard. However, if they do rank within the top 83 teams, this will open up an additional spot in the ranking order for a team ranked 84th. The TBD teams in the pilot alliance will be determined during the draft event.

Figure: Pilot Alliance



Aus	TBD	TBD

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.

2.6 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.

2.7 ISS Final Competition

The top 13 alliances on the leaderboard at the end of semifinal play plus one wildcard alliance (a total of 14 alliances) will advance to the ISS Finals Competition. The wildcard Alliance will reserve a spot for the Australian pilot team. Alliances ranked below rank 14 will be invited to participate in the Virtual Finals. See Section 2.8 for details about the Virtual Finals.

The ISS finals will take place aboard the International Space Station with live transmission to MIT. All teams will be invited to live broadcast events at MIT (US) and ESTEC (EU.)

2.7.1 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 CO_2 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 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 match of their player, but finishing the competition will take priority.

To summarize, time priority will be allocated to:

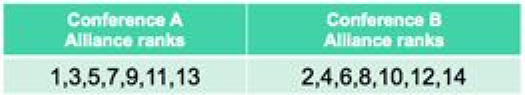
1) Running all submissions aboard the ISS at least once
2) Completing the tournament bracket
3) Running all submissions during live video

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

2.7.2 Competition Format

The alliances will be divided into 2 conferences for the ISS 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: Division of Teams between Conferences



Conference A Alliance ranks	Conference B Alliance ranks
1,3,5,7,9,11,13	2,4,6,8,10,12,14

Each conference will include one "bye" team (alliances ranked #1 and #2 automatically advance to the conference semi-finals) and 2 brackets of 3 alliances each. Each bracket will play 3 matches in round-robin style: alliance 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 ISS Competition Bracket figure.) The following rules determine the winner:

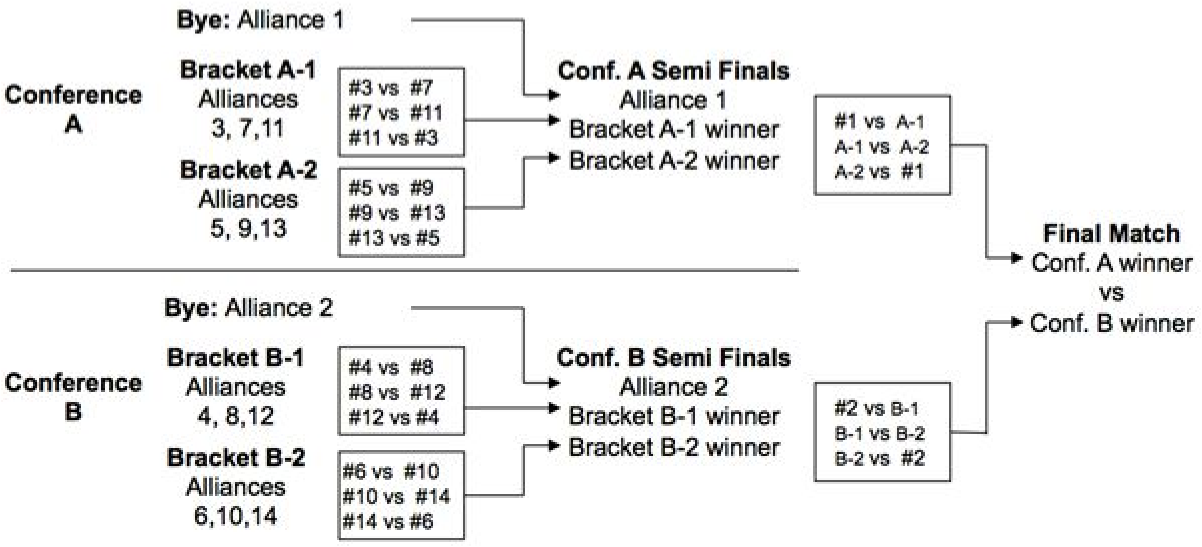
1. The alliance with the most wins advances
2. If alliances are tied for wins, the alliance with the highest total score advances
3. If scores are tied, simulation results will be used to break the tie

The semi-final match between the top 2 bracket winners and the "bye" team will also be played in round-robin style. The winner of this match is determined in the same way as the bracket winners:

1. The top 2 alliances with the most wins in their bracket, advance
2. If there is a tie for wins, the alliance(s) with the highest total score in their bracket advance
3. If scores are tied, simulation results will be used to break the tie

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

Figure: 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 CO_2 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.

2.7.3 Scoring Matches

Scores in the scoring matches will be determined according to these rules:

Case 1: Successful Match, Both Satellites Return Unique Score (e.g. 130, 151)
• The scores will be recorded as the official score for the match.

Case 2: Either Satellite Returns an Invalid Score (e.g. 130, 255)
• If the first run of a match is not successful, the match will be rerun, time permitting.
• If the second run of a match is not successful, the results from a simulated match will be used.

2.8 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 TBD teams will advance to the Virtual Finalist competition. The final format of the Virtual Finalist competition will depend on the number of teams participating in the Virtual Finals and will be announced as the tournament progresses. The Virtual Finalist competition will be run in simulation by the ZR staff. The results of the competition will remain secret until the ISS Final competition.

During the live broadcast of the ISS Finals the ZR Staff will replay the Virtual Finalist matches with commentary and will announce the results. *Time permitting, the final championship match of the Virtual Finalist competition will be competed aboard ISS.*

3. Season Rules

3.1 Tournament Rules

All participants in the Zero Robotics High School Tournament 2015 must abide by these tournament rules:

- The Zero Robotics team (MIT / Top Coder / 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 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).

Revision History

Revision Number	Date	Changes Made
1.0.0	2015-09-12	Release Version
1.1.0	2015-09-16	Clarifying how mirror items work.
1.2.0	2015-09-23	Added and clarified missing information in Section 1. Please also note the modified Ethics Code in Section 3.2.
2.0.0	2015-10-02	3D Manual Update.
2.1.0	2015-10-08	Clarifications regarding uploading, as well as fixing some potentially confusing typos.
3.0.0	2015-11-05	Alliance Manual Update
3.1.0	2015-11-12	Fixed instances of “Mirrored Across X” to “Mirrored Across Y” + a clarification about tiebreakers.