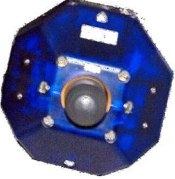




ZERO  ITALY

R  B  TICS

HS TOURNAMENT 2015

Game Manual

Ver. 1.1.0



Zero Robotics High School Tournament 2015: SpySPHERES

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.

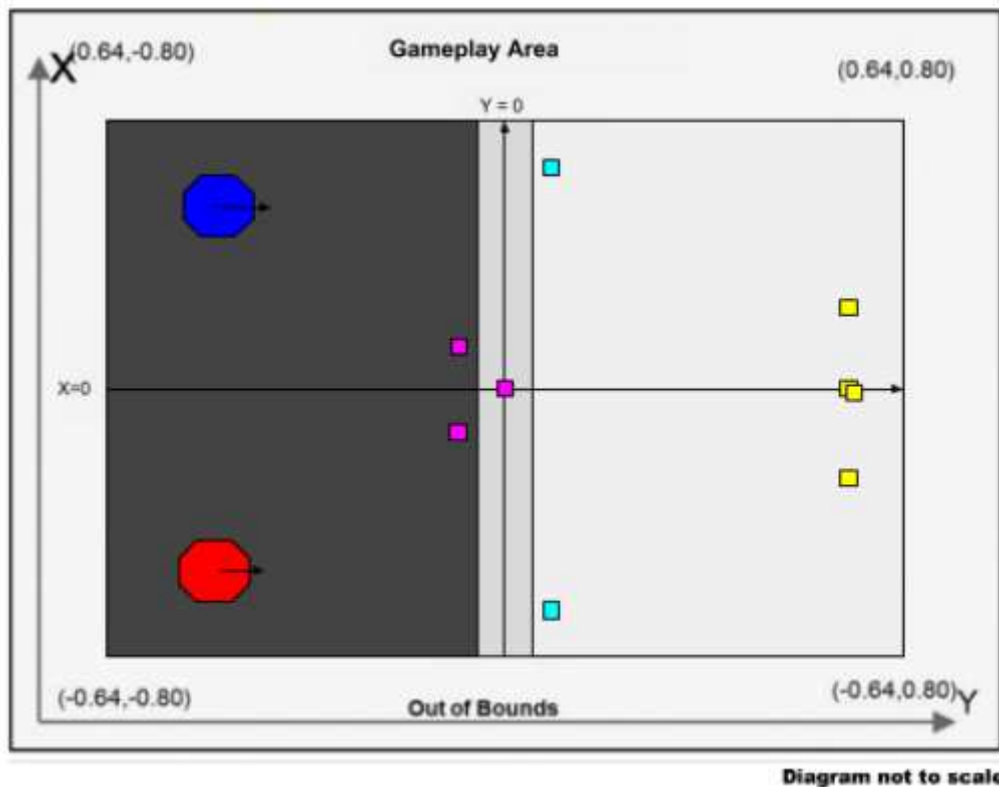
Let's get that tech!
Alvar Saenz-Otero
BLU CEO



ZRHS 2015 Game Manual

1. Game Overview

Figure: Game Overview



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.

Teams shall work on projects by choosing game **SpySPHERES 3D**



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, the satellites must be facing Earth, which is in the positive Z direction.

The Light, Dark, and Grey zones will switch 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

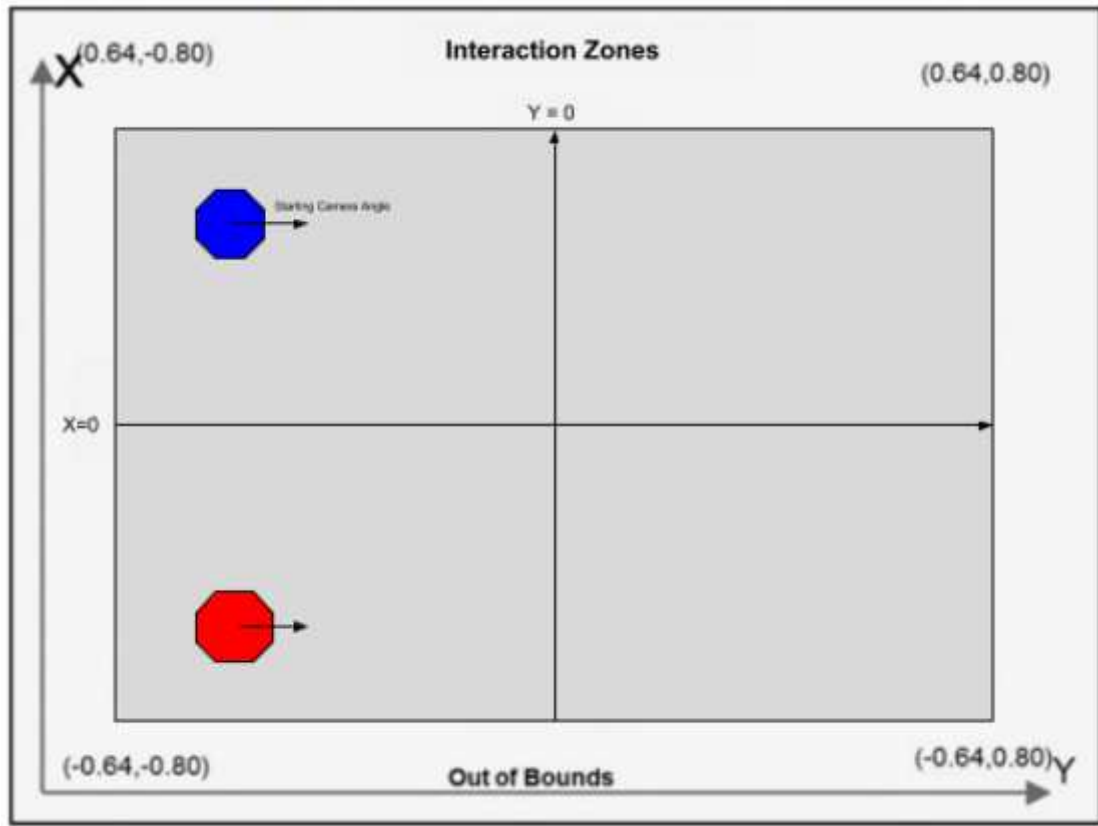


Diagram not to scale

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!



Mechanic Summary Table

Score Items	4
Mirror Items	Pictures are worth Negative Points
Energy Items	3
Light Zone	Switching
Uploading	Must be facing Positive Z
Out of Bounds	Light is the Same as In Bounds

1.2 Satellite

Each team will write the software to command a SPHERES satellite to move in order to complete the game tasks. A SPHERES satellite can move in all directions using its twelve thrusters. 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();           void sendMessage(unsigned char inputMsg);
int getMemoryFilled();             float getDarkGreyBoundary();
bool posInLight(float pos[]);      float getLightGreyBoundary();
bool posInDark(float pos[]);       float getLightInterfacePosition();
bool posInGrey(float pos[]);       unsigned char receiveMessage();
```



```

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 getItemLoc(float pos[], int itemID);
int getCurrentTime();
bool isFacingOther();
float getEnergy();
float getOtherEnergy();
int getItemType(int itemID);
int getNumItem();
int getNumMirrorsHeld();
bool useMirror();

```

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()`.

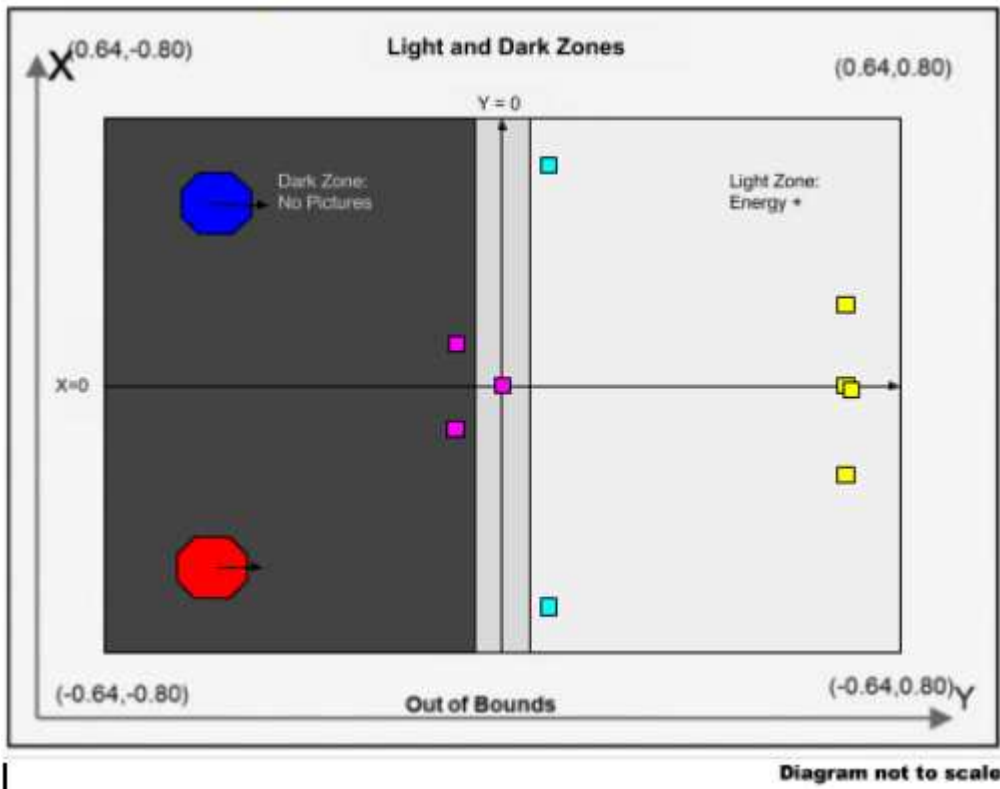


1.3.2 Light and Dark Zones

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



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.

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 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

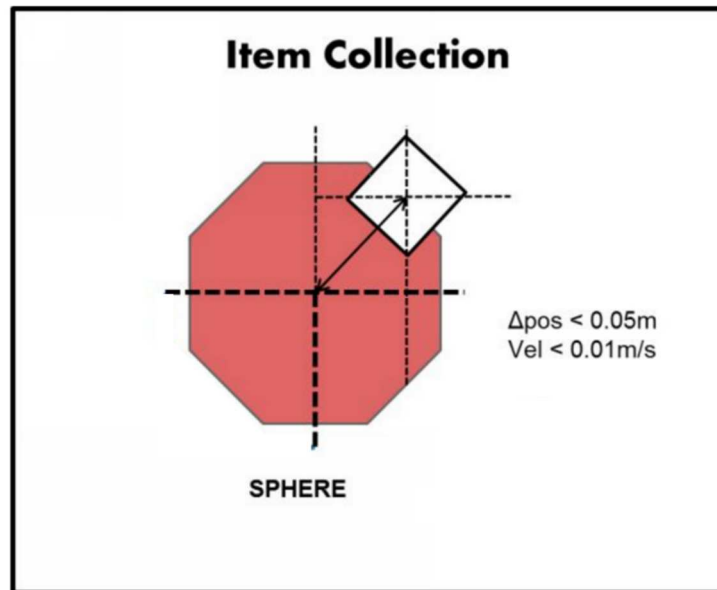


Figure: Item Locations & Types

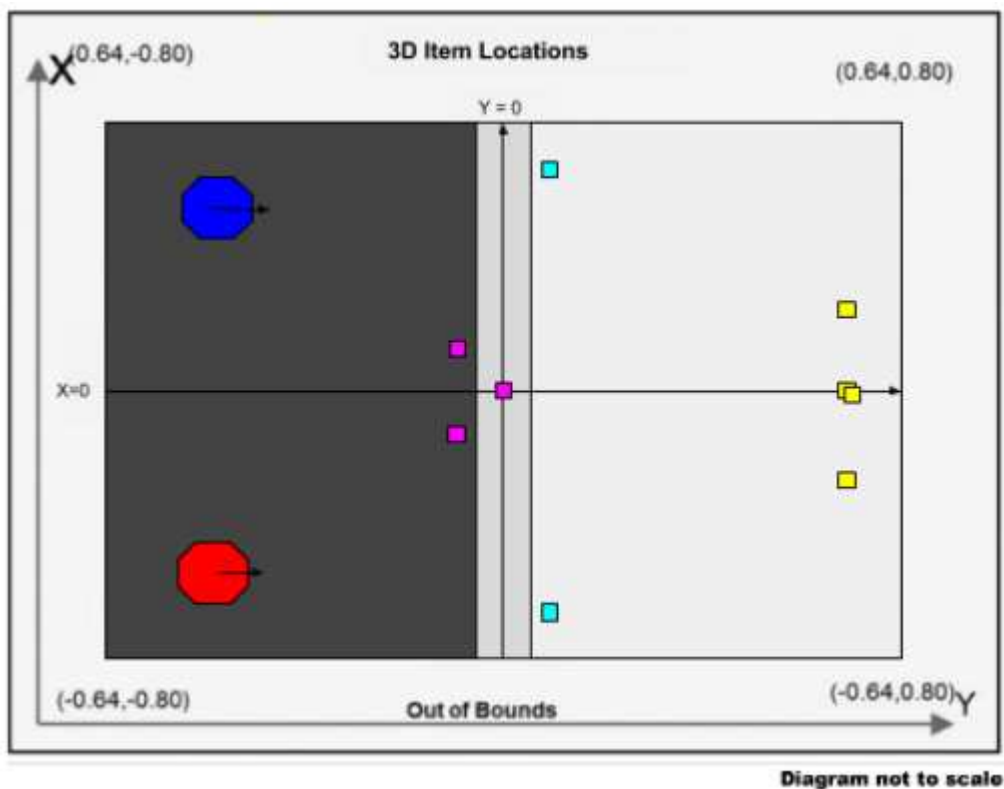


Table of Item Locations:

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

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:

$$\text{points} = 2.0 + 0.1 / (\text{distance} - \text{PHOTO_MIN_DISTANCE} + 0.1)$$

Where PHOTO_MIN_DISTANCE is 0.5. If the opposing satellite is using an active mirror, the amount of points the picture is the opposite of the above value.

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

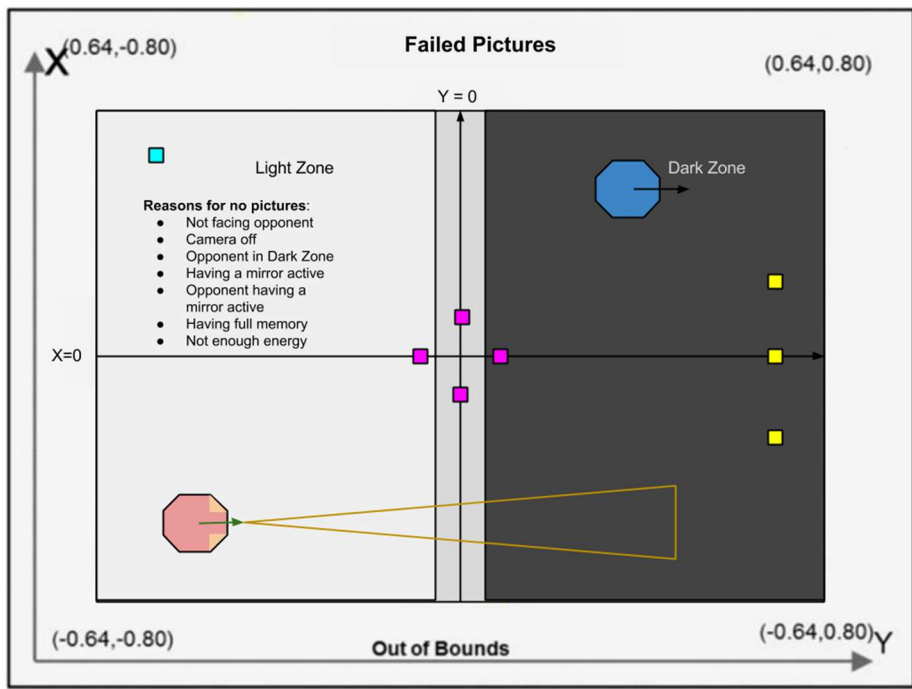
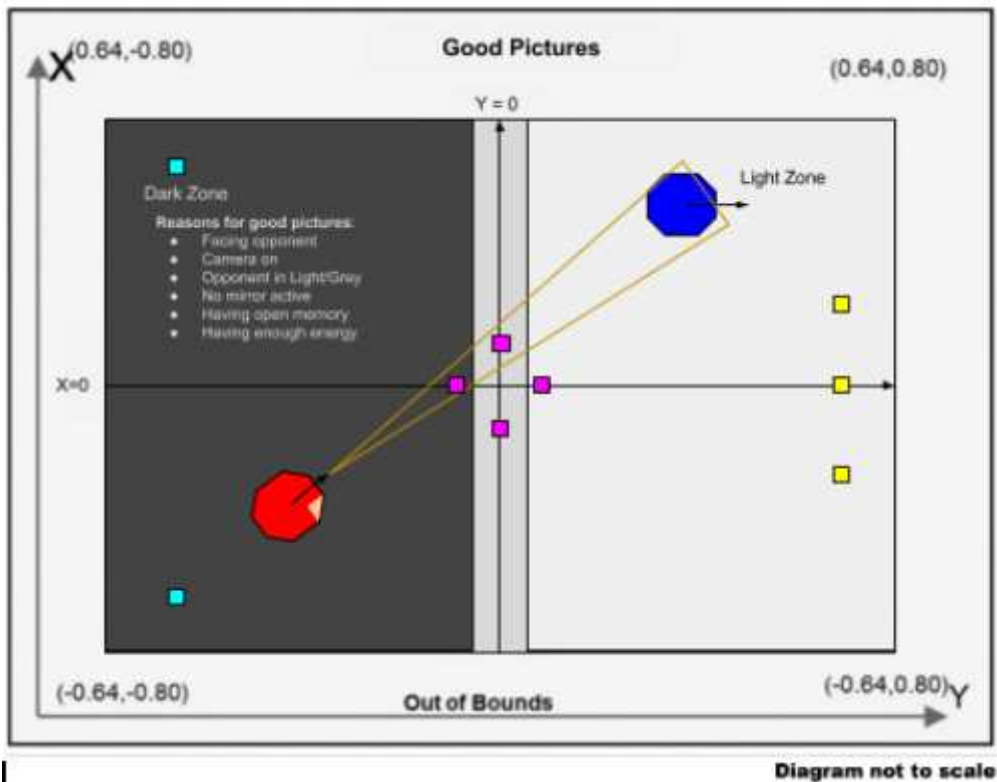


Diagram not to scale



Figure: Conditions for a Successful Picture



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, 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	$2.0 + \frac{0.1}{\text{distance} - 0.5 + 0.1}$ per picture without a mirror, and the opposite of that per picture with a mirror.
Score Items	1.5 per item

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

Oct 30 (Fri)	Kick-off
Dec 11 (Fri)	Semifinal Competition Deadline
Jan 15 (Fri)	Final Code Due
End Jan	Final Competition

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 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 10 teams above you and the 10 teams below you. If there are less than 10 teams above you, you will play all teams above you plus the 10 below you. If there are less than 10 teams below you, you will play all the teams below you and the 10 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”.

2.1.6 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 R_0 in the equation for the probability of your team beating another payer of rating R_1 in a single match.

$$\text{Probability}(W) = e^{R_0} / (e^{R_0} + e^{R_1})$$

The variable R_0 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.

2.7 Final Competition

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

The Finals will take place in simulation in Torino (I). All teams will be invited to live broadcast events at Politecnico di Torino.

2.7.1 Overview and Objectives

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 Finals.

2.7.2 Competition Format

The teams will be divided into 2 bracket for the Finals. 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

Bracket A	Bracket B
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)

The following rules determine the winner:

1. The team with the most wins advances
2. If teams are tied for wins, the team with the highest total score advances
3. If scores are tied, another simulation will be run between the tied teams

Then the two winners for each bracket will play against each other for the 1st and 2nd place; in case of tie a second simulation will be run by changing the colors.

Then the two second teams for each bracket will play against each other for the 3rd and 4th place; in case of tie a second simulation will be run by changing the colors.



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.



Revision History

Revision Number	Date	Changes Made
1.0.0	2015-10-29	Release Version based on 2.1.0 from 3D MIT

