ZERO ROBOTICS ISS PROGRAMMING CHALLENGE
HIGH-SCHOOL TOURNAMENT 2012:

Rapid Entry Trajectory by Retrograde ablatiOn (RETRO)

GAME MANUAL
V3.1.0

Dr. Alvar Saenz-Otero

Space Systems Laboratory
37-381

Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139

http://zerorobotics.mit.edu

2012-Sep-08

To: Zero Robotics Teams

Re: Rapid Entry Trajectory by Retrograde blation (RETRO) program

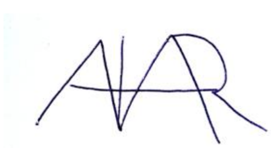
Attention to all teams:

The gradually accumulating cloud of space debris in low-Earth orbit threatens humanity's ability to launch new space missions. The area around Earth is full of hazards, ranging from tiny flecks of metal to huge school bus-sized dead satellites. The debris poses a danger to astronauts, functioning satellites and space launches. Right now, each piece must be carefully catalogued and avoided. If we wait too long, collisions between the debris may cause the situation to cascade and spiral out of control, a phenomenon known as Kessler Syndrome. It could make space exploration and the use of satellites out of reach of humanity for hundreds of years.

Enter the Rapid Entry Trajectory by Retrograde blation (RETRO) program. NASA and DARPA have directed MIT to create two companies: BluSpace and RedSpace. These companies will compete in a challenging demonstration mission with a lucrative debris removal contract at stake. Both companies are using specially outfitted RetroSPHERES satellites launched into a polar orbit to deploy micro dust clouds that can deorbit small pieces of space debris with high velocity collisions (ablation).

After deploying the dust, the RetroSPHERES must rendezvous with two decommissioned polar satellites, repurposing parts from their communications antennas for dust sensing. To enhance the challenge while alleviating launch costs, several resupply packs have been pre-positioned as secondary payloads on previous commercial launches. The packs provide additional propulsion and debris removal capabilities. Finally, to ensure each satellite can navigate an unknown space debris field, the demonstration requires the competitors to de-orbit while passing through their opponent's micro dust zone. Accurate sensing and navigating in this space is critical.

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



Alvar Saenz-Otero

MIT SPHERES Lead Scientist

Contents

1 Game Play

Matches are played between two SPHERES satellites that compete to get through a virtual course fastest using the least fuel. The game is broken up into three zones. In the first zone, both SPHERES have the opportunity to lay down virtual dust clouds in the playing field, to simulate space junk ablation. In addition, the dust clouds are placed in the "return" path of the opponents when they head to the their finish.. In the second zone, the player must rendezvous with a virtual disabled satellite. After acquiring the antenna of this virtual satellite, players may collect virtual re-supply packs that give the SPHERES satellite extra fuel and charge. The antenna enables the SPHERES satellite to detect the dust clouds laid out by the other player; they have to navigate through the dust clouds to reach the finish line. If a player enters a dust cloud, they will be slowed down (and therefore consume more time and fuel). Players can use charge (obtained in the re-supply packs) to boost their ability to identify the location and size of a dust cloud or to shrink a dust cloud. Players must complete the course within three minutes. The fastest player with the most fuel at the end of the match is the winner!

2012-10-06: Updates for the 3D game will be highlighted in yellow. Values in the tables that have not changed will not be highlighted. The full set of document changes can be found in the Document Control Panel at the end of the manual.

2012-10-30: Updates for the alliance phase will be highlighted in blue. Values in the tables that have not changed will not be highlighted. The full set of document changes can be found in the Document Control Panel at the end of the manual.

1.1 Game Layout

The Zero Robotics High School Tournament 2012 begins with competitions in simulation. The competition mimics the operational volume available aboard the International Space Station, where the 3D finals will be conducted in January 2013. For the 2D game, the arena is a plane with only X and Y cardinal dimensions. The game arena encompasses the complete area where the SPHERES satellites can operate. However, the game is played in a smaller area called the *Interaction Zone*. If players leave the Interaction Zone they may still be within the arena operational area, but they will be considered out of bounds.

```
API: call game.getCurrentPhase to determine which zone you are in
```

The dimensions of the Interaction Zone are

Table 1 Interaction Zone Dimensions

	2D	3D	Alliance
X [m]	[-0.64 : +0.64]	[-0.64 : +0.64]	[-0.64 : +0.64]
Y [m]	[-0.80 : +0.80]	[-0.80 : +0.80]	[-0.80 : +0.80]
Z[m]	n/a	[-0.65 : +0.65]	[-0.65 : +0.65]

The *Interaction Zone* is divided into three game zones as presented in Figure 1. The players must pass through each zone, in sequence, in order to complete the game. Each zone is described in detail below. The dimensions of the zones are:

Table 2 Zone Dimensions: Blue

	2D	3D	Alliance
Zone 1			
X [m]	[0.00 : +0.64]	[0.00 : +0.64]	[0.00 : +0.64]
Y [m]	[-0.80 : +0.10]	[-0.80 : +0.10]	[-0.80 : +0.10]
Z[m]	n/a	[-0.65 : +0.65]	[-0.65 : +0.65]
Zone 2			
X [m]	[-0.64 : +0.64]	[-0.64 : +0.64]	[-0.64 : +0.64]

Y [m]	[+0.10 : +0.80]	[+0.10 : +0.80]	[+0.10 : +0.80]
Z[m]	n/a	[-0.65 : +0.65]	[-0.65 : +0.65]
Zone 3			
X [m]	[-0.64: 0.00]	[-0.64: 0.00]	[-0.64: 0.00]
Y [m]	[-0.80 : +0.10]	[-0.80 : +0.10]	[-0.80 : +0.10]
Z[m]	n/a	[-0.65 : +0.65]	[-0.65 : +0.65]

Table 3 Zone Dimensions: Red

	2D	3D	Alliance
Zone 1			
X [m]	[-0.64 : 0.00]	[-0.64 : 0.00]	[-0.64 : 0.00]
Y [m]	[-0.80 : +0.10]	[-0.80 : +0.10]	[-0.80 : +0.10]
Z [m]	n/a	[-0.65 : +0.65]	[-0.65 : +0.65]
Zone 2			
X [m]	[-0.64 : +0.64]	[-0.64 : +0.64]	[-0.64 : +0.64]
Y [m]	[+0.10 : +0.80]	[+0.10 : +0.80]	[+0.10 : +0.80]
Z [m]	n/a	[-0.65 : +0.65]	[-0.65 : +0.65]
Zone 3			
X [m]	[0.00 : +0.64]	[0.00 : +0.64]	[0.00 : +0.64]
Y [m]	[-0.80 : +0.10]	[-0.80 : +0.10]	[-0.80 : +0.10]
Z [m]	n/a	[-0.65 : +0.65]	[-0.65 : +0.65]

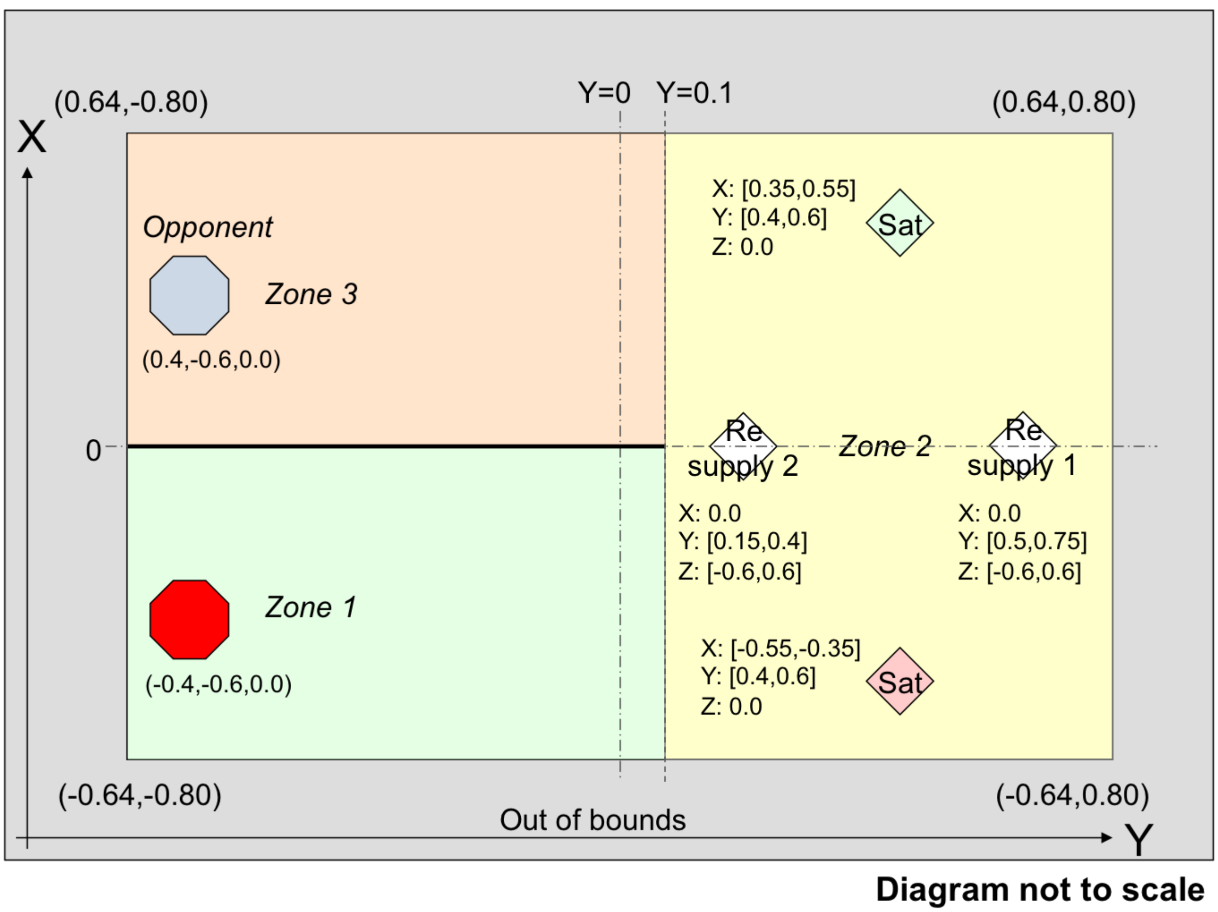


Figure 1 Game Overview

1.2 Satellite

Each team will write the software to command a SPHERES satellite to move in order to complete the game tasks. Each SPHERES satellite is capable of moving in all directions via its set of twelve thrusters. The actual SPHERES satellite, like any other spacecraft, has a fuel tank (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 the players must adhere to wisely use real SPHERES resources.

API: the satellite is primarily controlled through the following functions:

- api.setPositionTarget: use MIT's algorithm to move to a position
- api.setVelocityTarget: use MIT's algorithm to set velocities
- api.SetForces: command the thrusters to translate the satellite
- api.setAttitudeTarget: rotate the satellite to point in a direction
- api.setTorques: command the thrusters to rotate the satellite
- api.getMyZRState: determine your position and orientation
- api.getOtherZRState: get the position and orientation of the other satellite

1.2.1 Time

Each player has a maximum 180s in order to complete a match. The game will automatically timeout (and the final score will be applied) after this period.

1.2.2 Fuel

In the game each player is assigned a *virtual fuel* allocation. For RetroSPHERES the initial amount depends on the phase of the tournament as presented in Table 4. The allocation is the total sum of fuel used in seconds of individual thruster firing. Once the allocation is consumed, the game activity ends for that player. Players have the ability to obtain additional fuel time by picking up a Re-supply spacecraft (see "Items" below).

Table 4 Initial Fuel Allocation

	2D	3D	Alliance
Initial Fuel Allocation [s]	50s	50s	50s

The virtual fuel allocation is consumed for both physical and game-play reasons, these include:

- any motion initiated by player
  - translation or rotation
  - acceleration (to start moving) and deceleration (to stop)
- any motion required by the satellite to account for game-play violations
  - firing when out of bounds (see "Out of Bounds" below)
  - maneuvers to avoid collisions (see "Zone 2" below)
- going out of bound triggers an additional fuel penalty (see "Out of Bounds" below)
- carrying dust cloud material into Zone 2 (see "Zone 2" below)

- entering an opponent's dust cloud (see "Zone 3" below)

The most important incentive to conserve virtual fuel allocation is that the game score is primarily based on fuel remaining at the end of a match (see "Scoring" below).

```
API: call game.getFuelRemaining to determine how much fuel you have left
```

1.2.3 Charge

In the game each player starts with one (1) unit of charge. Charge is used to boost the ability to determine the location of dust clouds or to shrink their size. Additional charge can be acquired by collecting one or two re-supply packs (see "Items" below).

The amount of charge available per iteration of `loop()` is described in Table 5:

Table 5 Charge use limit cycle

	2D	3D	Alliance
Charge per `loop()`	Unlimited	Number of dust clouds created	Number of dust clouds created

In the 3D game, players can use the same number of units of charge per second while in Zones 2 and 3 as the number of dust clouds they created in Zone 1. Players who create one dust cloud can use one unit of charge per second; two dust clouds, two units of charge per second; etc.

```
API: call game.getCharge to determine how many charge units you have left
```

1.2.4 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:

Table 6 Inter-satellite communications bandwidth

	2D	3D	Alliance
Message size	unsigned short	unsigned short	unsigned short

Note: bandwidth allocation may either increase or decrease as the tournament progresses.

1.2.5 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 Zone 1: Deploying Dust Clouds

Zone 1 is strictly limited to the dimensions presented in Table 2 & Table 3. Leaving the Zone 1 region immediately triggers one of two results:

- Entering Zone 2 permanently changes game zones (regardless of velocity or rotation)

- Leaving in other directions induces an "out of bounds" penalty (see below)

The SPHERES satellite is deployed at:

Table 7 SPHERES Satellite Deployment Locations

	2D	3D	Alliance
Red			
X [m]	-0.40	-0.40	-0.40
Y [m]	-0.60	-0.60	-0.60
Z[m]	n/a	0.00	0.00
Blue			
X [m]	+0.40	+0.40	+0.40
Y [m]	-0.60	-0.60	-0.60
Z[m]	n/a	0.00	0.00

From that initial position, the objective of this zone is for your satellite to deploy virtual dust clouds (see example in Figure 2). The initial dust volume each player starts off with is given below.

Table 8 Initial Dust Volume

	2D	3D	Alliance
Initial Dust Volume	$0.21m^3$	$0.42m^3$	$0.42m^3$

You can deploy dust clouds anywhere within the zone, with two restrictions on their sizes and proximity. To initiate cloud creation the satellite center must be at least 0.05m from the edge of all other clouds, and dust clouds will stop expanding if their radii exceed 0.3m in the 3D game or 0.35m in the 2D game, or if they come within 0.01m of another dust cloud, as illustrated in Figure 2. If a dust cloud extends beyond the zone boundaries, it will continue to affect a SPHERES satellite, even if the SPHERES satellite is out of bounds.

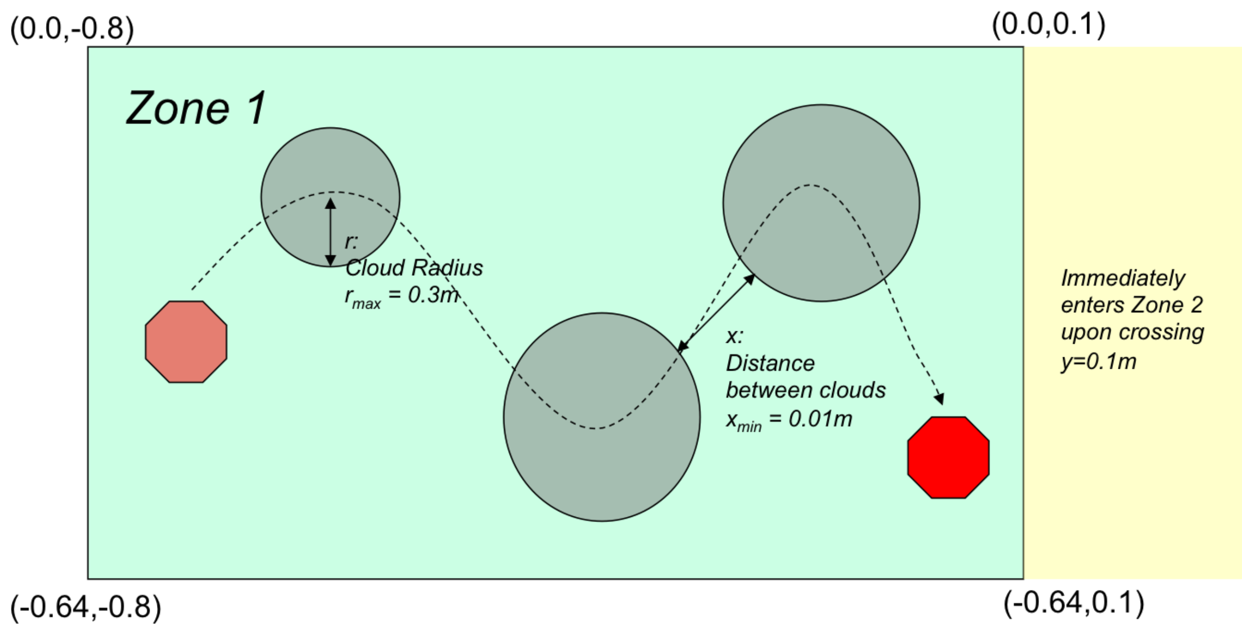


Figure 2 Zone 1: Deploying Dust Clouds

1.3.1 Creating Dust Clouds

The creation of dust clouds depends on your availability of dust material, the motion of the SPHERES satellite, and the location of any previously created clouds. The dust clouds are spherical. There is a maximum of ten (10) dust clouds per player. The steps to create a dust cloud are as follows:

1. Initiate obstacle creation through the API function `game.startObstacle`, which returns true if the satellite has material remaining and obstacle creation is not currently underway. There is no constraint on initial velocity or angular velocity.
2. Rotate about any axis to attain an attitude at the desired angle from the satellite's initial attitude at the call to `game.startObstacle`.
   - Dust cloud creation is governed by the equation

$$vol = k \cdot \theta,$$

   where θ is the angle between the satellite's initial and final attitudes, measured in radians, and k is the dust cloud creation constant in Table 9.

Table 9 Dust Cloud Creation Constant

	2D	3D	Alliance
k (m^3/rad)	N/A	N/A	0.036

3. Stop (velocity must be less than 0.01m/s and angular velocity must be less than 2.3°/s) at the center where the dust cloud is to be created.
4. Drop the created dust cloud through the API function `game.stopObstacle`, which returns true and drops the created dust cloud if the velocity and angular velocity constraints are met.
   - The dust cloud will automatically be resized if necessary such that the surface of the dust cloud is not within 0.01m of the surface of another dust cloud.
   - The dust cloud will automatically be resized if its volume exceeds the amount of material remaining.
   - If the velocity and angular velocity conditions are not met, `game.stopObstacle` will return false, and obstacle creation will remain underway.
   - No material is used until the dust cloud is successfully dropped.

Table 10 Maximum Dust Cloud Radius

	2D	3D	Alliance
$radius_{max}$ (m)	0.35	0.3	0.3

The process and restrictions of creating a cloud are illustrated in Figure 3 Cloud Creation process and restrictions..

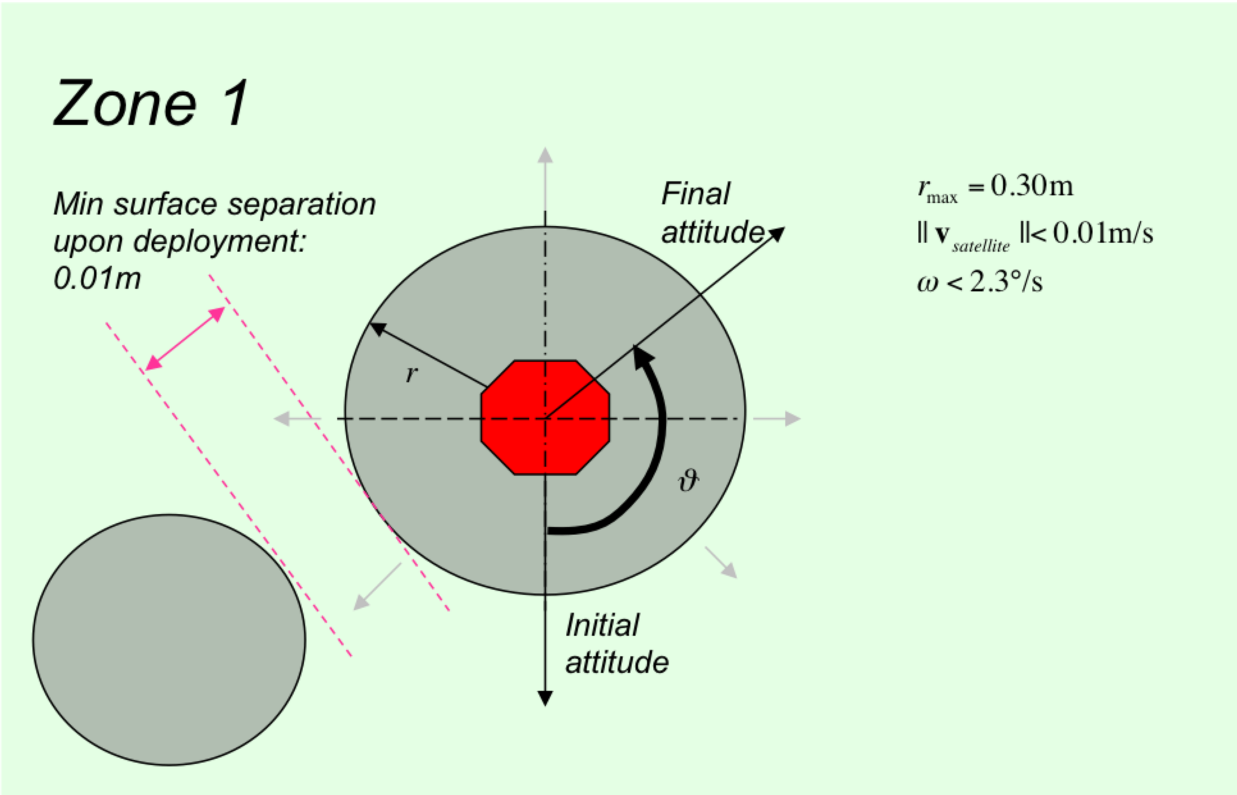


Figure 3 Cloud Creation process and restrictions.

```
API:  call game.startObstacle to start creating a cloud
      call game.getCurrentObstacleSize to determine the radius of the dust
            cloud under creation
      call game.getRemainingMaterial to see how much dust remains
      call game.stopObstacle to stop creating a cloud
```

SCORING ALERT:

Having dust remaining in your satellite when you enter Zone 2 results in a fuel penalty as follows:

Fuel penalty = $Dust_{remaining}$ / $Dust_{total}$ * Material Penalty

Table 11 Material Penalty

	2D	3D	Alliance
Material Penalty	16.875s	12.0s	12.0s

Moreover, to initiate cloud creation the satellite center must be at least 0.05m from the edge of all other clouds.

1.4 Zone 2: Item Collection

Zone 2 contains the disabled satellite you *must* rendezvous with in order to collects its antenna. The zone also contains the re-supply spacecraft which you *may* want to rendezvous with in order to increase fuel and charge.

When operating within Zone 2 you might come in close proximity of your opponent. If the RetroSPHERES code detects an impending collision with the other SPHERES satellite it will activate its own overriding control, changing the satellite motion and charging you for the fuel usage to complete the avoidance maneuver.

SCORING ALERT:
If collision avoidance is activated, you will be charged the fuel used to avoid the collision.

1.4.1 Disabled Spacecraft

In order to exit this phase of the game and continue to Zone 3 you ***must*** rendezvous with the disabled satellite in your path. If you attempt to move into Zone 3 before you dock with the disabled satellite, that motion will be considered out of bounds and penalized accordingly (see "Out of Bounds" below). The disabled satellite location will vary during each match. In the 3D game, the Red's disabled spacecraft will be at the image of the position of Blue's disabled spacecraft after a 180° rotation about the y-axis. The position of the satellite is between:

Table 12 Disabled Spacecraft Deployment Locations

	2D	3D	Alliance
Red			
X [m]	[-0.55 : -0.40]	[-0.55 : -0.35]	[-0.55 : -0.35]
Y [m]	[+0.30 : +0.65]	[+0.40 : +0.60]	[+0.40 : +0.60]
Z[m]	n/a	[-0.60 : +0.60]	0.00
Blue			
X [m]	[+0.40 : +0.55]	[+0.35 : +0.55]	[+0.35 : +0.55]
Y [m]	[+0.30 : +0.65]	[+0.40 : +0.60]	[+0.40 : +0.60]
Z[m]	n/a	[-0.60 : +0.60]	0.00

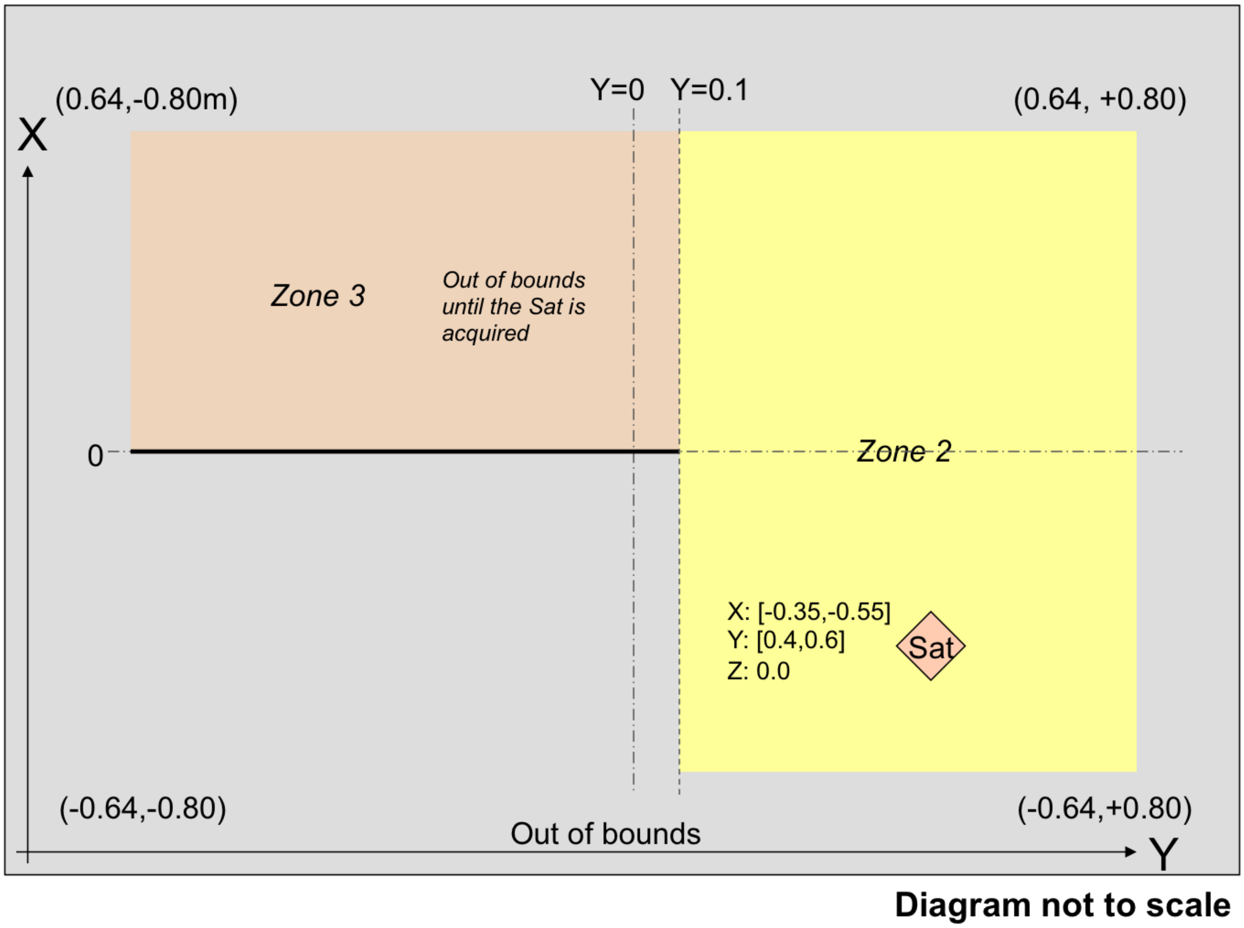


Figure 4 Zone 2: everything else is out of bounds until the disabled satellite is picked up.

NOTE: even though there is an identical disabled satellite for the other player, it is not accessible to your satellite: it effectively does not exist for your player.

Acquisition of the disabled satellite antenna happens once you have completed the docking maneuver (see below). Once you have the antenna you can start to use it to detect dust clouds in Zone 3.

In the 2D game, you must rendezvous with the disabled satellite before being able to pick up any re-supply items. In the 3D game, you may pick up items in any order, but you must rendezvous with the disabled satellite to make Zone 3 accessible.

1.4.2 Re-supply Spacecraft

Two re-supply spacecraft were deployed in the area exactly between the RedSpace and BluSpace areas. Their locations will always be equidistant from the disabled satellites and from the SPHERES satellites' deployment locations. The deployment locations of the two re-supply spacecraft are:

Table 13 Re-supply spacecraft deployment locations

Axis	2D	3D	Alliance
Re-supply 1			
X [m]	0.00	0.0	0.0
Y [m]	0.55 to 0.79	[+0.5 : +0.75]	[+0.5 : +0.75]
Z [m]	n/a	0.0	[-0.60 : 0.60]

Re-supply 2			
X [m]	0.00	0.0	0.0
Y [m]	0.15 to 0.40	[+0.15 : +0.40]	[+0.15 : +0.40]
Z [m[	n/a	0.0	[-0.60 : 0.60]

Both RedSpace and BluSpace SPHERES satellites can collect the re-supply spacecraft; only the first player to reach the re-supply will be able to benefit from it. Once a re-supply spacecraft has been used, the other player cannot benefit from it.

The re-supply spacecraft provide both fuel and charge. The charge is constant. However, the available fuel benefit depends on the percentage of dust remaining in your SPHERES satellite when you crossed into Zone 2, as presented in Table 15. The re-supply spacecraft provide:

Table 14 Re-supply spacecraft benefits

Axis	2D	3D	Alliance
Re-supply 1			
Fuel	17s	17s	17s
Charge	4 units	4 units	4 units
Re-supply 2			
Fuel	12s	12s	12s
Charge	6 units	6 units	6 units

Your ability to rendezvous and benefit from these items depends on the amount of dust remaining in your satellite once you enter Zone 2 as shown in Table 15.

Table 15 Dust remaining effect on ability to collect items

Dust remaining into Zone 2	# items	Re-supply fuel worth
100%	3	100%
25% < $Dust_{remaining}$% < 100%	2	Fuel_resupply = (0.55 + 0.45*$Dust_{remaining}$ / $Dust_{total}$) * Item_fuel
< 25%	1	0% (cannot pick up resupply)

1.4.3 Picking up an item

In order to dock to an item, you need to perform a specific maneuver (pictured in Figure 5). The steps to collect an item are:

1. Determine where items are located.
    - The satellite is equipped with a sensor that can measure the distance between the satellite and an item with limited accuracy (see Table 16 Sensor Accuracy). Use the API function `game.pingForItems`, available from the start of the game, to find the distance from the satellite to each item to the nearest multiple of 3cm—the sensor's smallest unit of measurement.

Table 16 Sensor Accuracy

	2D	3D	Alliance
Sensor Accuracy	N/A	N/A	+/- 1.5cm

2. Position the satellite within 0.05m of the item's center.
    - The satellite's velocity must be less than 0.01m/s.

- The satellite's angular velocity must start at less than 2.3°/s.

3. Rotate the satellite >90° along any axis. Do not attempt to rotate faster than 80°/s.

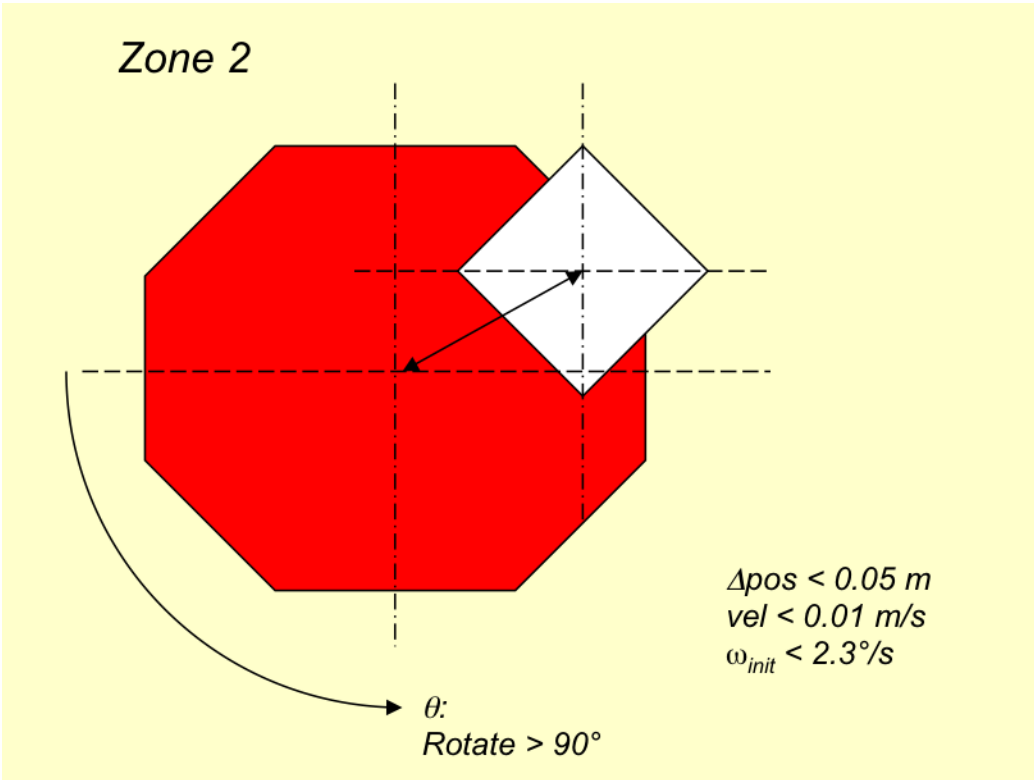


Figure 5 Maneuver to rendezvous with an item

API:	use game.pingForItems to determine the distance to each item. Use game.haveObject to determine if you successfully picked up an item. Object 0 = disabled satellites Object 1 = Re-supply 1 Object 2 = Re-supply 2 use game.otherHasObject to determine object of the other player.

1.5 Zone 3: Path to Finish

The objective of Zone 3 is to navigate across the finish line. Once you enter Zone 3, exiting the zone 3 boundaries is considered an "Out of Bounds" violation, you may not re-enter Zones 2 or 1.

To get to the finish, you will need to fly through your opponent's Zone 1, where they may have deployed dust clouds (Figure 6). The dust clouds will slow you down, costing you both time (going slower) and fuel (to overcome the effects of the cloud). To avoid the dust clouds you must use data from the antenna to detect them. You may use charge to extend the range of the antenna detector and/or to shrink any dust clouds in your path.

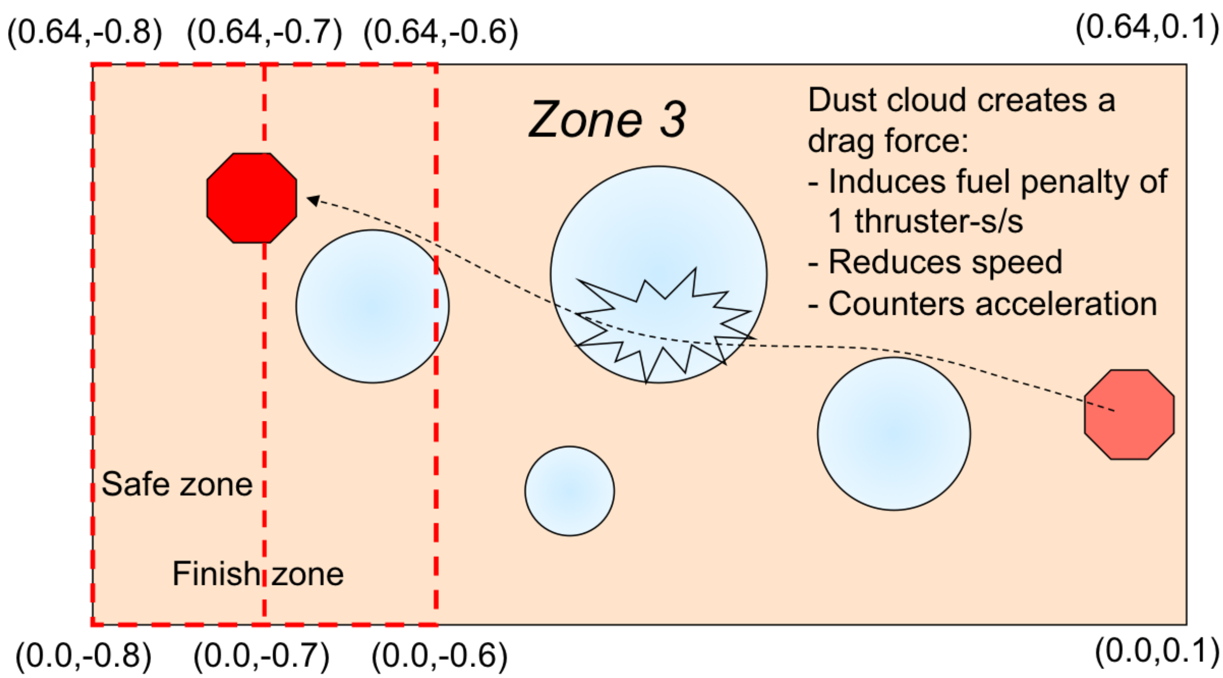


Figure 6 Zone 3: Path to finish

1.5.1 Dust Clouds

A dust cloud will affect the motion of your satellite if the center of the SPHERES satellite enters within the radius of the dust cloud. The clouds are of equal density regardless of their size, therefore all clouds affect your satellite in the same way. Entering a cloud creates a *drag force* proportional to the square of the velocity of your spacecraft:

$$\mathbf{F}_{drag} = -(250\ \mathrm{kg/m}) \cdot \mathbf{v}_{satellite} \cdot \| \mathbf{v}_{satellite} \|$$

In the 3D game, entering a dust cloud also induces a fuel penalty of 1 thruster-second / second while your satellite remains within the cloud. In the 2D game, there is no fuel penalty for entering an opponent's dust cloud. In the Alliance phase of the competition, there is a safe zone behind Y=-0.7m where the satellite is unaffected by dust clouds: if the satellite has a Y-coordinate less than -0.7m, the drag force and fuel penalty do not apply.

Table 17 Dust Cloud Penalty

	2D	3D	Alliance
Dust Cloud Penalty (thruster-s / s)	0	1	1

If your SPHERES satellite has sufficient charge and you have detected a dust cloud (and saved its ID), you can use the API function `game.shrinkObstacle` to decrease the radius of the dust cloud by 10% (the shrinking applies immediately after the call). In the 3D game, you may only shrink obstacles that are currently visible. Note that your use of charge is limited in the 3D game (see Table 5).

API: `game.shrinkObstacle` shrinks the radius of the identified dust cloud by 10% (if charge is available)

1.5.2 Dust Cloud Gravity

In the Alliance phase, the opponent's dust clouds exert a gravitational force on the satellite. When the satellite lies outside of the dust cloud's surface, the dust cloud exerts the following force on the satellite:

$$\mathbf{F} = G \cdot m_{satellite} \cdot m_{cloud} \cdot \frac{(\mathbf{r}_{cloud} - \mathbf{r}_{satellite})}{\| \mathbf{r}_{cloud} - \mathbf{r}_{satellite} \|^3},$$

where G is the universal gravitational constant (6.67300×10^{-11} m^3 kg^{-1} s^{-2}) and $\mathbf{r}_{cloud}$ and $\mathbf{r}_{satellite}$ are the positions of the dust cloud and the satellite respectively. The mass of the cloud can be calculated assuming a density of 2.68×10^8 kg m^{-3}. Note that the mass of the satellite is 4.3 kg.

When the satellite is located inside the dust cloud, gravity still acts toward the center of the cloud. The force exerted on the satellite is

$$\mathbf{F} = G \cdot m_{satellite} \cdot m_{cloud} \cdot (\mathbf{r}_{cloud} - \mathbf{r}_{satellite}).$$

If the satellite has a Y-coordinate less than -0.7m, dust cloud gravity does not affect the satellite.

1.5.3 Detection Cone

The antenna collected from the Zone 2 disabled spacecraft enables a radar feature in your SPHERES satellite. The radar has a limited detection cone as presented in Figure 7. The detection cone is ±12° in angle. The distance of the cone defaults to 0.4m (dark cone). The detection distance can be extended beyond the *interaction zone* by using charge. Each use of `extendView` will increase the range during that `loop()` cycle.

An obstacle is in view if any part of the obstacle is within the detection cone. To determine if any obstacles are in view, use the API function `game.getIdentifiedObstacles`, which populates an array with the list of all dust clouds that have been identified. A visible obstacle will not be identified unless the function is called.

API:	`game.getIdentifiedObstacles` populates an array with all identified obstacles **NOTE: <u>must</u> pass an array of <u>exactly TEN</u> (10) Obstacles as the parameter**

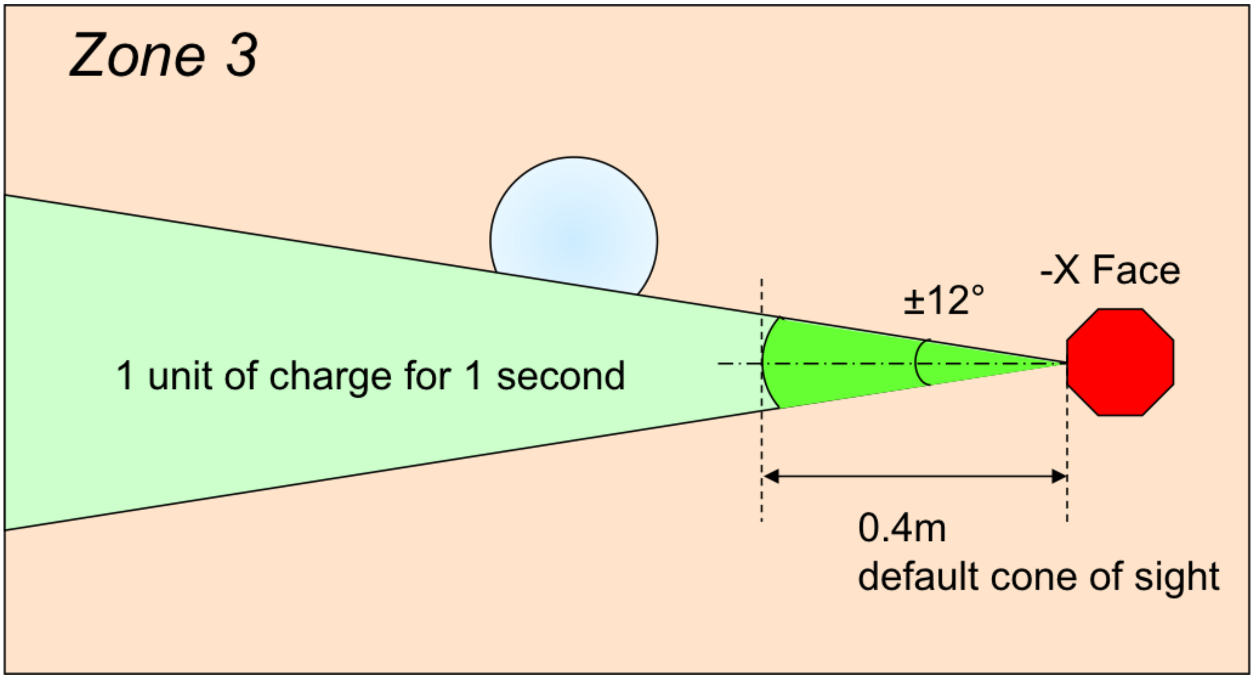


Figure 7 Detection cone (standard: dark green); extended (light green)

1.5.4 End of Activity

Your activity (and therefore time and fuel to calculate scoring) finishes when any one of the following conditions occurs:

- Your SPHERES satellites crosses into the finish zone and completes the "finish maneuver":

Table 18 Finish Maneuvers

	2D	3D	Alliance
Maneuver	Cross the Y = -0.6m line.	1. Cross the Y = -0.6m plane. 2. Cross the Z = 0m plane while in the Finish Zone. 3. Attain a distance of 0.3m from the Z = 0m plane while in the Finish Zone.	1. Cross the Y = -0.6m plane. 2. Cross the Z = 0m plane while in the Finish Zone. 3. Attain a distance of 0.3m from the Z = 0m plane while in the Finish Zone.

In 3D, the finish maneuver entails entering the Finish Zone (Y < -0.6m), crossing Z = 0m, then attaining a Z-coordinate with |Z| > 0.3m, by traveling in either direction from Z = 0m. A player who enters the Finish Zone with a positive Z-coordinate will have to attain a negative Z-coordinate and then cross either the Z = 0.3m plane or the Z = -0.3m plane.

Or

- You run out of time (180s), or
- You run out of allocated fuel.

SCORING ALERT:

The "fuel remaining" and "time to finish" used for scoring are the values stored at the instant that any one of these conditions is met.

1.6 Out of Bounds

You must remain within the boundaries of each zone until you meet the requirements to continue to the next zone:

Zone 1 to Zone 2	Cross Y > 0.1m
Zone 2 to Zone 3	Rendezvous with the disabled spacecraft, then enter Zone 3 crossing Y < 0.1 and Red X > 0 Blue X < 0
Zone 3 to Finish Zone	Cross Y < -0.6m

If you exit out of bounds, the MIT code will override your commands and force the satellite to stop its motion in the direction that would continue to push it out of bounds (other directions are not affected). The fuel used to stop this motion will be charged to your fuel usage.

Additionally, whenever the SPHERES satellite is out of bounds, you will be penalized as follows:

- Penalty = 2.5 * (seconds out of bounds)

SCORING ALERT:

Being out of bounds results in both fuel usage and a penalty beyond the actual fuel used while out of bounds:

- The SPHERES satellite will stop motion in the direction out of bounds
- Fuel penalty = 2.5 * (seconds out of bounds)

2 Scoring

Scoring depends on four main factors:

1. You must have entered Zone 3
   - If you do not the game will count as a loss regardless of the other player's score
2. Fuel remaining at the time your activity finished
3. The time when you crossed the finish line,

   or

   Location at match end if you did not cross the finish line
4. Which player finished first

The final score for the match is calculated as follows:

Table 19 Score Calculation

Step			2D points	3D points	Alliance points
1	Did **not** enter Zone 3		-100s (Loss)	-100s (Loss)	-100s (Loss)
2	**Fuel** at end of activity	f	2 * (fuel_remaining)	2 * (fuel_remaining)	2 * (fuel_remaining)
3	Player with lower finish **time** gets a bonus*	t	higher_time – lower_time *max: 17*s	higher_time – lower_time *max: 17.75s*	higher_time – lower_time *max: 17.75s*

4	If player did not finish, penalty in **distance** from finish	d	50s/m * (distance from Y= -0.6m)	50s/m * (minimum necessary distance to travel to complete finish maneuver)	50s/m * (minimum necessary distance to travel to complete finish maneuver)
5	Determine total score: • if 1 • if 2 only (not 3, not 4) • if 2+3 • if 2+4		-100s f f + t f - d	-100s f f + t f - d	-100s f f + t f - d
6	The score is normalized by multiplying by 23/100				

* *Note: if the other player does not finish, their "higher_time" is 180s.*

3 Tournament

A Zero Robotics Tournament consists of several phases called Competitions. The following table lists the key deadlines for the 2012 tournament season:

Table 20 Tournament Key Dates

Date (2012)	Event
Sep 8	Kick-off webcast
Sep 28	Registration Deadline
Oct 5	2D Simulation Competition Deadline
Oct 12-14	2D Ground Demonstrations
Oct 29	3D Simulation Competition Deadline
Nov 3	Alliance Formation Event
Dec 1	Semi-Final Simulation Competition Deadline
Dec 16	ISS Code Due
Jan TBD, 2013	ISS Final Competition

The rankings of each *Competition* are determined through the use of a "Leaderboard". The 2D Simulation Competition is *not* an elimination round; everyone advances to the 3D Competition. At the end of the 3D Simulation Competition an elimination round takes place, with the top 54 US / 36 EU ranking teams moving to the Alliance phase of the Tournament. The top 54 US / 36 EU teams will then form alliances of three (3) teams each: 18 US and 12 EU alliances will be formed. The alliances will compete against each other, with the top 9 US and 6 EU alliances moving to the ISS Final Competition.

3.1 The Leaderboard

This year's tournament introduces a new format for the Competition phases with a continuously updated ranking system called *The Leaderboard*. The Leaderboard uses a system similar to the Elo rating system for chess players called Whole History Ranking[1] as well as ideas from the *TrueSkill*®[2] rating system used for the Xbox Live gaming platform. The Leaderboard tracks all matches a team has played against other players in the course of the competition and creates a score based on the outcomes. At the end of each competition phase, the final standings on the Leaderboard will determine the points awarded toward the tournament standings.

[1] R. Coulom, "Whole-history rating: A bayesian rating system for players of time-varying strength", Computers and games, 2008.

[2] R. Herbrich and T. Minka, "TrueSkill: A Bayesian Skill Rating System", in Advances in Neural Information 20, 2007.

3.1.1 How Ranks are Calculated

In any match between two teams A and B, the probability of a match outcome is estimated as a function of the ranks of the two teams:

$$P_{Awins} = \frac{e^{R_A}}{e^{R_A} + e^{R_B}}$$

where R_A is the rank of Team A, and R_B is the rank of Team B. Players of nearly equivalent ranks have roughly equal probability of winning a match, while differences in rank result in one of the teams being more heavily favored. After a match is played, a correction is applied to the rank to account for the new match outcome. There are several factors that affect a team's rank:

- **Match Outcomes:** A team that consistently wins matches will usually have a higher rank.
- **Opponent Rank** Winning against a higher ranked team will usually improve a team's rank.
- **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.
- **Time** Over the course of a short period of time, the rank is not expected to change drastically, so new match outcomes are not as heavily weighted. Over longer periods of time, new match outcomes are more heavily weighted.
- **Uncertainty** In addition to the rank R, the Leaderboard tracks a measure of uncertainty. It is assumed a team's rank will change over the competition period but it is not known in which direction, so uncertainty grows with time. New matches help to decrease uncertainty by providing more performance data.

The score calculated based on these factors is the estimate of the team's rank (mean) minus the uncertainty (3-sigma).

The Leaderboard calculates rankings in real-time (shortly after submission) from the beginning of a competition and until the submission deadline. All matches during the competition period count towards the ranking in the competition. The Leaderboard scores are cleared at the start of each competition (and at the end of "warm-up" periods, see below).

3.1.2 Playing Matches

To play a match the team must use the Submissions tool on the Team Management page. A team may make any number of submissions up until the "final submission period" (see below). **Important:** be sure to commit your code before submitting it to the leaderboard. The most recently committed copy will be used, not your local copy.

Submitting the project will automatically queue twenty (20) matches between the submitted player and the twenty (20) teams with the nearest mean score (not necessarily the published ranking). A randomized set of ten (10) of the twenty (20) matches will be as the Blue sphere; the other ten, as the Red sphere. The most recent submission will be used for scoring on the Leaderboard until a new one replaces it.

You can review the simulation and results of the matches from the most recent submission, including any matches run against you when other teams make submissions. When you submit a new project, you will no longer be able to see the results from previous submissions. *Note: all participants can see all the simulations from the most recent submission of all teams.*

3.1.3 Competition Periods

Each competition consists of three main periods: warm-up, regular, and final submission:

- **Warm-up** The warm-up period is the period when teams can test their submissions with no impact on their final ranking. The warm-up period consists of the first seven (7) days from the start of the competition.

- **Regular** During the regular period teams may make as many submissions as they want, understanding that all matches played with those submissions will affect their ranking.
- **Final Submission** In the last 24 hours before the competition deadline the total number of project submissions will be limited. The matches played with these submissions will affect the ranking.

Table 21 Competition Periods

	2D	3D	Alliance
Warm-Up	YES	YES	YES
Regular Leaderboard	YES	YES	YES
Final Submission Limit	10	10	10

3.1.4 Ranking Results

While the Leaderboard processes rankings in real-time for individual submissions, it requires a global update after a large number of submissions. Therefore final rankings will be announced up to 24 hours after close of a competition. The Leaderboard may not be available during this 24-hour period.

3.2 2D Simulation Competition

All teams that complete a valid registration are eligible to participate in the 2D Simulation Competition.

3.3 3D Simulation Competition

All teams that complete a valid registration are eligible to participate in the 3D Simulation Competition.

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

3.4 Alliance Formation Event

The top 54 US / 36 EU teams will form 18 US / 12 EU alliances of three (3) teams each that will be working cooperatively to complete the semi-finals and, if not eliminated, the finals.

The ranking of the teams will be determined by combining their 2D Simulation and 3D Simulation Competitions scores:

$$Score = 0.25 * 2D\_NormalizedScore + 0.75 * 3D\_NormalizedScore$$

where

$$NormalizedScore = \frac{Score - MinimumScoreInCompetition}{MaximumScoreInCompetition - MinimumScoreInCompetition}$$

After the rankings for semi-finalist teams are announced, teams will have three days to contact each other about their 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.

Alliances will be formed during a *draft day* where teams will be able to contact each other and accept their position in the alliance. The alliances selection process will follow a *serpentine* pattern (illustrated in Figure 8):

- Team Rank 1 selects their partner from anyone between Rank 10 and Rank 54
- Team Rank 2 selects their partner from the remaining Rank 10 – 54.

- Up to when the first 18 pairs are created
- 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 18 teams
- The "2nd lowest" rank pair make the next selection
- Continue until all 18 alliances are formed

Note for EU: Form 12 pairs, then discuss, then complete 12 3-team Alliances.

Note: US and EU teams will be ranked in the same process, but will be split into two new groups for Alliance creation purposes.

Rule:

US: The 3 teams in an alliance cannot be from the same city/metropolitan area.

EU: The 3 teams in an alliance cannot be from the same country.

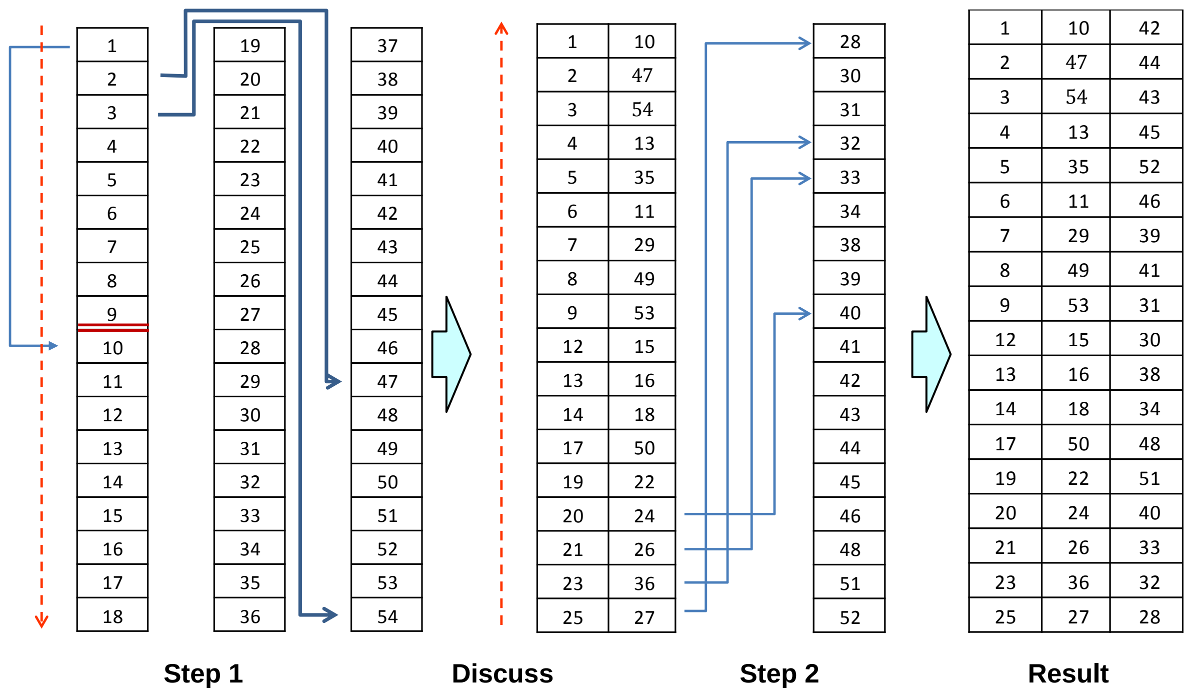


Figure 8 Alliance Creation Process

3.5 *Semi-Final Simulation Competition*

The 18 US / 12 EU alliances will participate in the Semi-Final Simulation competition.

When the Semi-Final 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 towards the project submissions.

3.6 ISS Final Competition

The top 9 US / 6 EU alliances in the Leaderboard will advance to the ISS Finals Competition. The 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).

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

3.6.2 Competition Format

The alliances will be divided into 3 US / 2 EU brackets of 3 alliances each (as shown for the US competition in Figure 9). Each bracket will play 3 matches in round-robin style: alliance A vs. B, B vs. C, and C vs. A.

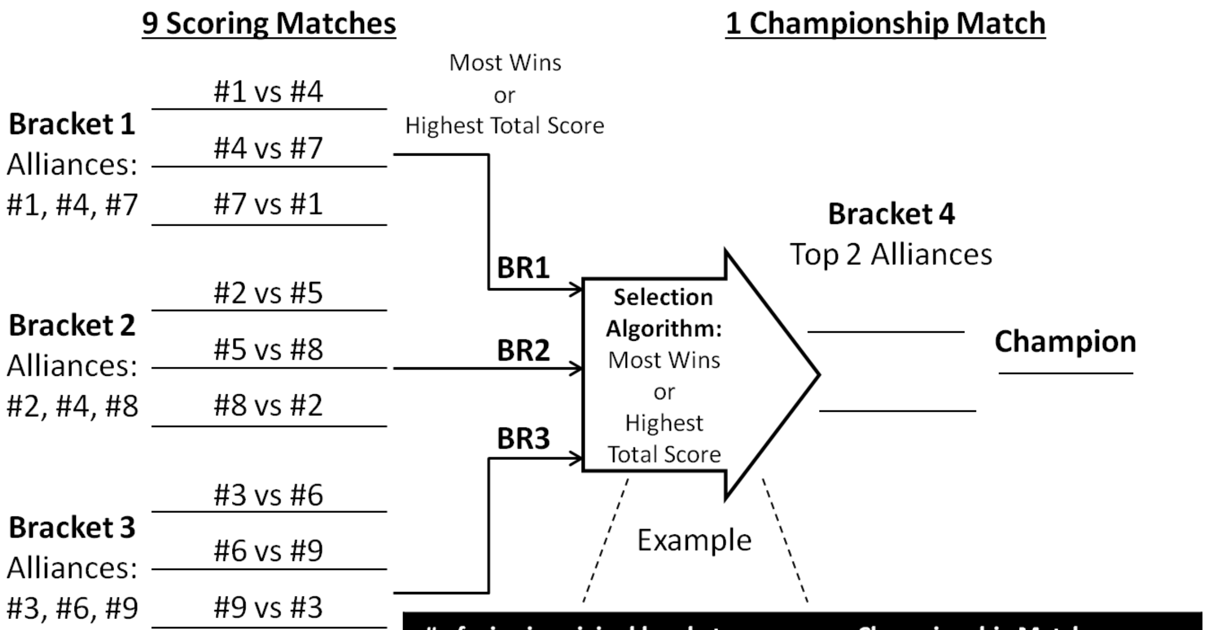
After the round-robins are complete, there will be a winner of each bracket (shown as BR1, BR2, BR3 in Figure 9) as determined by:

1. Any ties in Zone 2 will count as a loss for both teams
2. The alliance with the most wins advances
3. If alliances are tied for wins, the alliance with the highest total score advances
4. If scores are tied, simulation results will be used to break the tie

There will only be one championship match between between the top 2 bracket winners from the scoring matches. In the EU championship, there will be two winners that move straight to the championship match. For the US, the top 2 alliances are 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 of teams in the final match is the Zero Robotics Champion. The losing alliance will be awarded 2^{nd} place, and the last place bracket winner will be awarded 3^{rd} place.



# of wins in original bracket			Championship Match
BR1	BR2	BR3	
2	2	2	Top 2 scorers in original bracket
2	**2**	1	BR1 & BR2
2	1	1	BR1 & top scorer between BR2 and BR 3
1	1	1	Top 2 scorers in original bracket

Figure 9: US Competition Bracket

3.6.3 Definitions

Definition: *Successful Match*

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

Definition: *Simulated Match*

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

3.6.4 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: Successful Match, Satellites Tie (e.g. 130, 131)

- Both teams will receive 0 points (double loss)

Case 3: Either Satellite Returns an Invalid Score (e.g. 130, 255)

- If the first run of a match is not successful, the match will be re-run (time permitting)
- If the second run of a match is not successful, the results from a simulated match will be used

4 Season Rules

4.1 Tournament Rules

All participants in the Zero Robotics High School Tournament 2012 must abide by these tournament rules:

1. The Zero Robotics team (MIT / Top Coder / Aurora) can use/reproduce/publish any submitted code.
2. 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.
3. Teams are expected to report all bugs as soon as they are found.
   3.1. A "bug" is defined as a contradiction between the intent of the game and behavior of the game.
   3.2. The intent of the game shall override the behavior of any bugs up to code freeze.
   3.3. Teams should report bugs through the online support tools. ZR reserves the right to post any bug reports to the public forums (ZR will work with the submitting team, if needed, to ensure that no team strategies are revealed).
4. Code and manual freeze will be in effect 3 days before the submission deadline of a competition.
   4.1. Within the code freeze period the code shall override all other materials, including the manual and intent.
   4.2. 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.
   4.3. The code is finalized at the ISS Final Competition freeze (unless there is a critical issue which will affect the final tournament, including lessons learned from ground hardware testing and simulation).
5. Game challenge additions and announcement of TBA values in the game manual may be based on lessons learned from earlier parts of the tournament.

4.2 Ethics Code

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

5 Zero Robotics API Reference

http://zerorobotics.mit.edu/ZRHS2012/api/index.html

6 Lists of Figures and Tables

6.1 List of Figures

6.2 List of Tables

7 Document Control Panel

File Name:	`RetroSPHERES Manual.doc`
File Location:	`MIT ZR SVN: \trunk\Documents\Game Manuals\ZR 2012\`

Revision	Date	Changes	By
1.0	2012/09/08	Initial Release.	alvarso@mit.edu
1.0.1	2012/09/09	• Corrected errors in axis directions • Fixed missing minus sign in formula for volume rate • Corrected lower threshold of 2 item limit: 25%, not 30% • Changed $Dust_{remaining}\%$ to $Dust_{remaining} / Dust_{total}$ and made coefficients consistent with code • Corrected sign error in final fuel penalty. Distance from Y = -0.6 not +0.6 • Corrected rotation rate for starting obstacle pickup: 2.3°/s, not 10°/s	jgkatz@mit.edu
1.0.2	2012/09/11	• Corrected volume rate formula. See this forum post.	jgkatz@mit.edu
1.0.3	2012/09/17	• Corrected satellite deployment location and item 0 (disabled sat) location	jgkatz@mit.edu
1.0.4	2012/09/18	• Renamed the description for Table 8.	prashanw@mit.edu
1.0.5	2012/09/21	• Corrected schedule to include date of finalist code submission deadline • Emphasized that projects must be committed before submitting to leaderboard • Corrected description of getObstacleSize(): gets radius not volume • Clarified figure for cloud creation to distinguish between w_{max} and the maximum rotation speed	jgkatz@mit.edu
1.0.6		• Clarified the rates of rotation for picking up an item: rotational speed must be less than 2.3 deg/s to *start* item pickup	jgkatz@mit.edu
2.0.0	2012/10/6	• Updated 3D interaction zone dimensions. • Updated 3D zone boundaries. • Updated 3D initial fuel allocation. • Added description of fuel penalty for entering an opponent's dust cloud. • Updated 3D charge use limit per cycle. • Updated 3D satellite deployment positions. • Updated 3D initial dust cloud volume. • Renamed 8 to Initial Dust Volume. • Clarified conditions for creating a dust cloud. • Updated 3D dust cloud volume expansion rate. • Updated 3D dust cloud maximum radius. • Fixed 2D dust cloud maximum radius – should be 0.35m not 0.4m. • Added Table 10 Maximum Dust Cloud Radius. • Added Table 11 Material Penalty.	vmayar@mit.edu

		• Clarified location of disabled spacecraft in the 3D game. • Updated 3D disabled spacecraft location. • Corrected 2D disabled spacecraft location – Red's item0 x-coordinate range was reversed. • Updated order in which spacecrafts can be picked up in the 3D game. • Updated 3D re-supply spacecraft starting positions. • Updated 3D re-supply spacecraft locations. • Clarified use of getItemLocation() – only specifies item location when in zone 2. • Updated 3D fuel penalty for entering an opponent's dust cloud. • Added Table 16 Dust Cloud Penalty • Clarified conditions for shrinking obstacles in 3D game – obstacle must be visible. • Updated 3D finish maneuver. • Updated 3D scoring weights. • Updated 3D competition periods. • Updated diagrams for 3D competition. • Updated method for choosing automatically queued matches upon submitting to the leaderboard – now 10 above and 10 below, 10 as SPH1 and 10 as SPH2.	
2.0.1	2012/10/6	• Updated finish maneuver for 3D to make symmetric for both players. • Clarified 3D description of penalty in distance to finish in Table 18.	vmayar@mit.edu
2.0.2	2012/10/10	• Changed 3D re-supply spacecraft deployment locations – now equidistant from SPHERES deployment locations. • Changed 3D disabled satellite deployment location – Z-bounds are now [-0.6,0.6]. • Updated Figures 1 and 4 to reflect above changes. • Clarified description of re-supply spacecraft deployment locations.	vmayar@mit.edu
2.0.3	2012/10/11	• Changed game.getVisibleObstacles() to game.getIdentifiedObstacles().	vmayar@mit.edu
3.0.0	2012/10/30	• Changed obstacle creation steps, equation, and table for the alliance phase. • Updated values in tables for the alliance phase. • Updated Figures 1, 3, and 4 to reflect above changes. • Added equations and description for dust cloud gravity in the alliance phase. • Updated equation for NormalizedScore. • Added description of pingForItems().	vmayar@mit.edu
3.0.1	2012/11/03	• Filled in TBA value for sensor accuracy	jgkatz@mit.edu
3.0.2	2012/11/07	• Added description of safe zone.	vmayar@mit.edu
3.0.3	2012/11/14	• Clarified description of sensor accuracy.	vmayar@mit.edu

3.1.0	2012/12/06	• Added rules for ISS competition	jgkatz@mit.edu