MARC
Millennium All-Island Robot Challenge

COMPETITION OVERVIEW

Intoduction

Contents

In the year 2100, a spaceship orbiting the planet Mars is hit by an asteroid and is minutes away from destruction. The spaceship is a biological research lab containing valuable experimental data. You are the only crew-person on the ship and manage to reach an escape capsule. You notice a utility robot near by and decide to re-program it to collect experimental data packages scattered throughout the ship and return to the escape capsule before its battery runs out.

MARC - Millennium All-island Robot Challenge is a programming competition aimed at promoting the software industry among school children and university students. Competitors are expected to design an algorithm to navigate spaceship using the robot's vision and implement the algorithm using an API available in all popular programming languages.

The competition is held for two age categories, under 18 and open. The competition duration is four months and includes a total prize money of one million rupees.

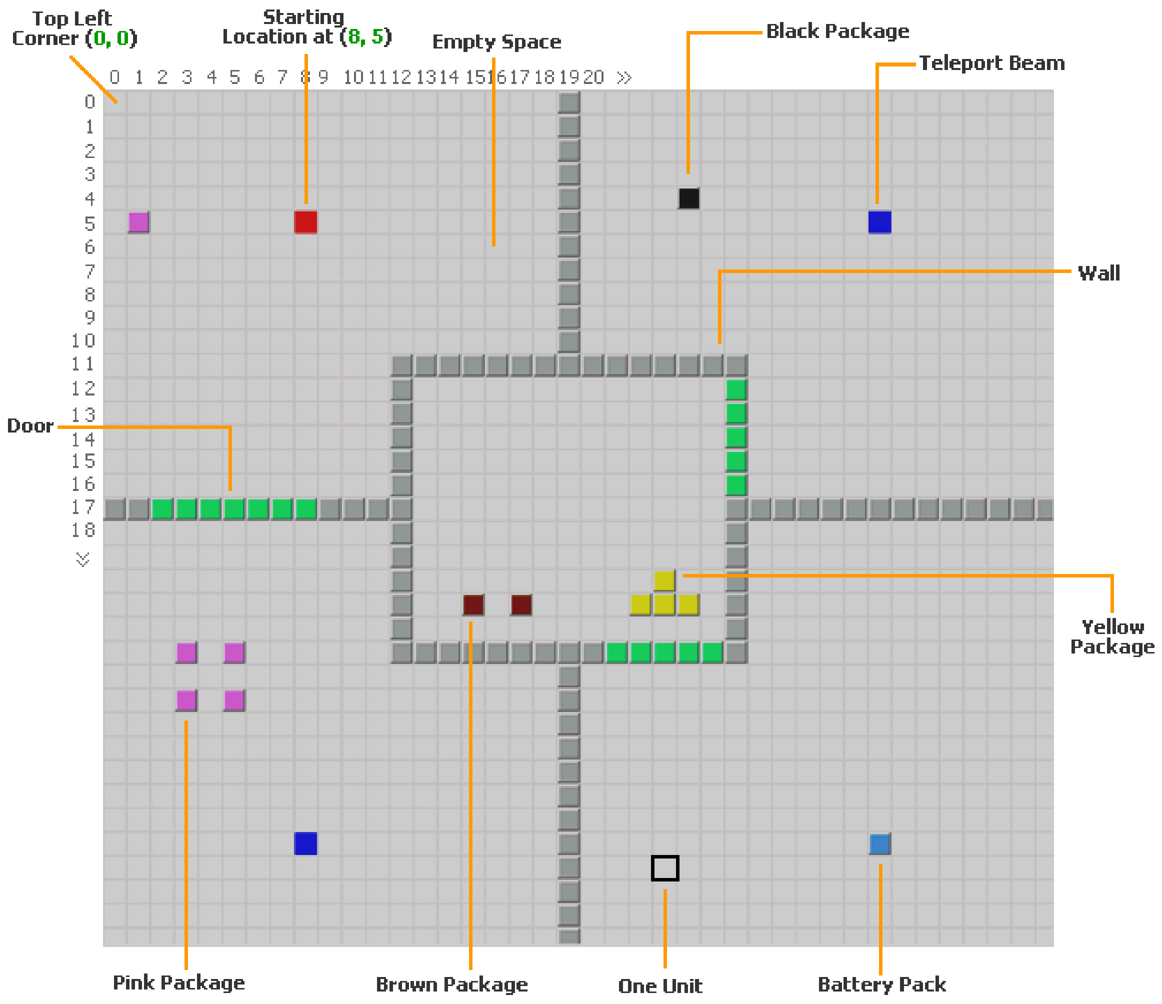
The remainder of this document describes the competition details and the development guidelines. You should read through this document before referring to any other documents supplied with the SDK. You may have to read this document several times as you get more familiar with the competition as information required to successfully complete the project is scattered throughout. Please be sure to take a look at the legal document (link will be available soon) containing all the competition rules. If you have further questions please refer to the FAQ section.

Spaceship

The spaceship is rectangular in shape and contains walls, doors, and teleport beams. The layout of the spaceship is mapped to a rectangular grid similar to graph paper. Locations of the spaceship is addressed using a (x, y) coordinate system where the top left hand corner is the (0, 0) position. The x coordinates increase from left to right and the y coordinates increase from top to bottom. The grid positions are a distance of one unit apart. That is, the distance between x=1 and x=2 is one unit and distance between y=1 and y=2 is also one unit.

Packages and battery packs are scattered around the spaceship and one location in the spaceship is

marked as the starting location which is where the escape capsule is located.
A two dimensional birds-eye representation of a spaceship and the grid is shown below.



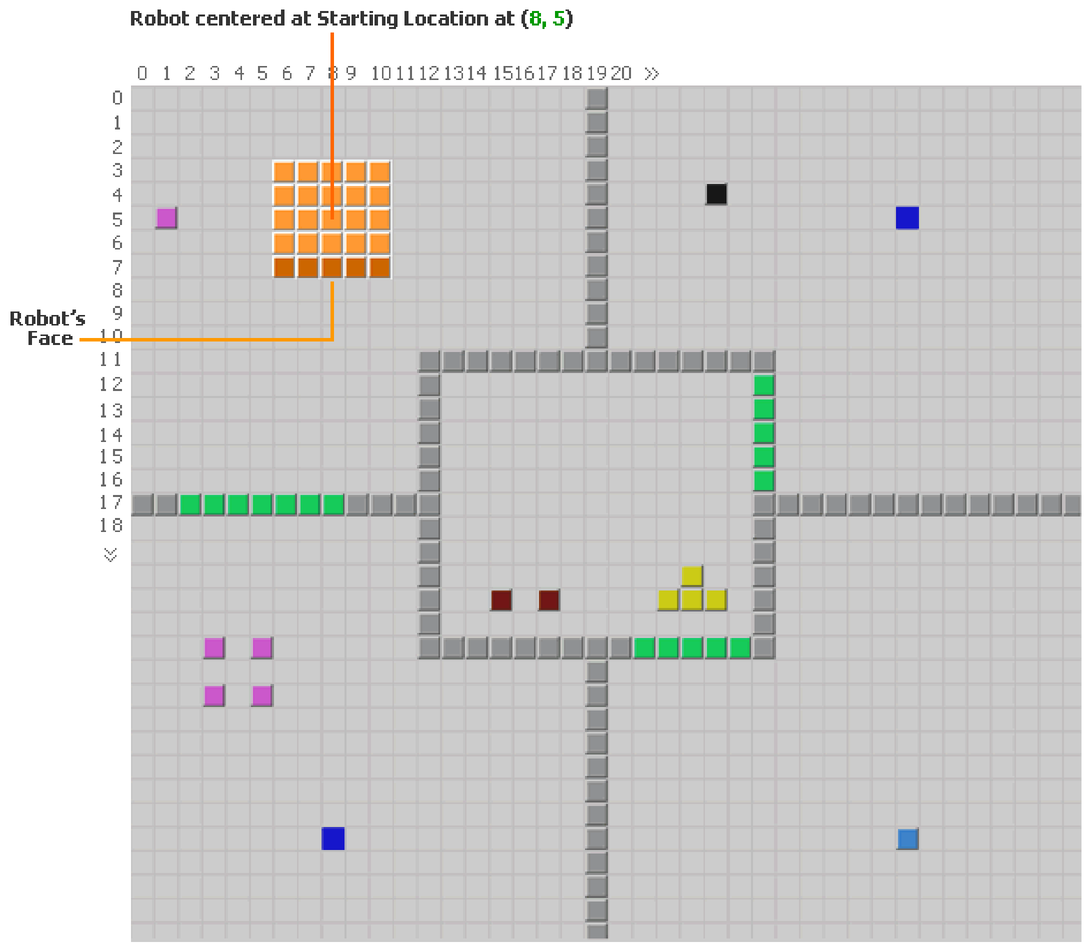
The walls of the spaceship are one unit thick and 5 units high. The doors of the spaceship are one unit thick and can vary in width from one unit to any length in multiples of units. A teleport beam is one unit long, one unit wide and 5 units high.

Spaceship.exe distributed with the SDK creates a virtual spaceship which can be used to test your robot. The *Shipbuilder.exe*, also provided with SDK, can be used to create spaceships of varying sizes and configuration.

Packages

There are four types of packages containing experimental data and samples. All packages are 1 unit wide, 1 unit long, and the height varies from 1 to 5 units. The packages are color coded. Yellow packages contain the most important experiments, followed by pink, and brown. Black packages are toxic and picking them causes the Robot to lose health.

Robot



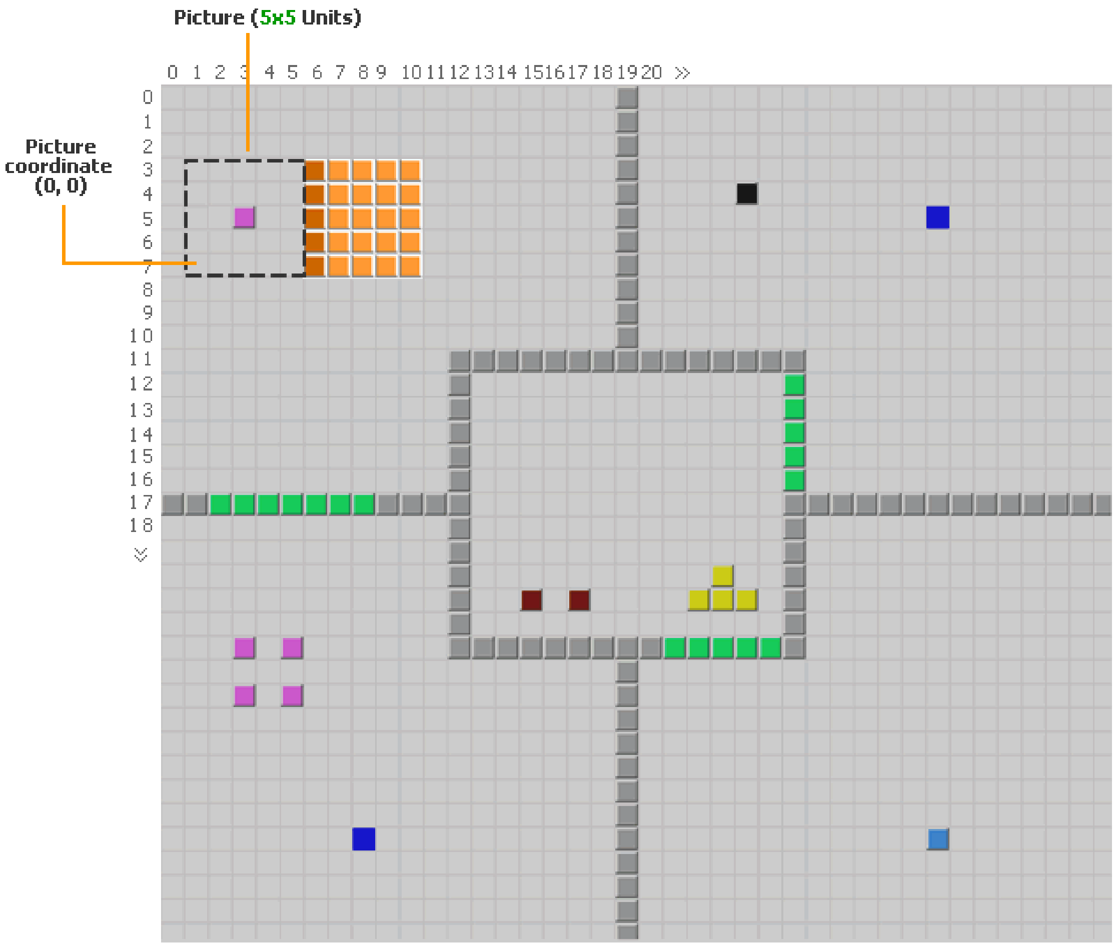
The Robot is a cube of 5 units long, 5 units wide and 5 units high as shown in the diagram above. One of its vertical sides is the face on which a special camera and a tractor beam are fixed. The Robot has a maximum storage capacity of 120 square units. That is the total height of all the packages stored cannot exceed 120. The Robot operates on battery power and all its actions consume battery.

The Robot can turn 90 degrees to the left or to the right relative to its face. It can also turn 180 degrees. The Robot can move any number of units in the direction in which it faces.

The Robot can take a picture in front of its face. The picture is 5 units wide and 5 units deep. The picture would record all the objects and their heights in the 5x5 area in front of the Robot's face. The Robot's special camera can see through walls, doors, and packages. If the Robot is near an edge of the Spaceship, the area outside would be recorded as walls in the picture.

The contents of a picture can be access through the (x, y) coordinate system similar to that of the spaceship. (0, 0) identifies the top left corner of the picture and (4, 4) identifies bottom right corner of the picture. By using the picture and the picture coordinate system the developer may determine the type of objects in front of the Robot and the height of the objects through the API. Note that the spaceship coordinate system and the picture coordinate systems are not the same.

The picture coordinate system is relative to the face of the Robot as shown in the diagram below.

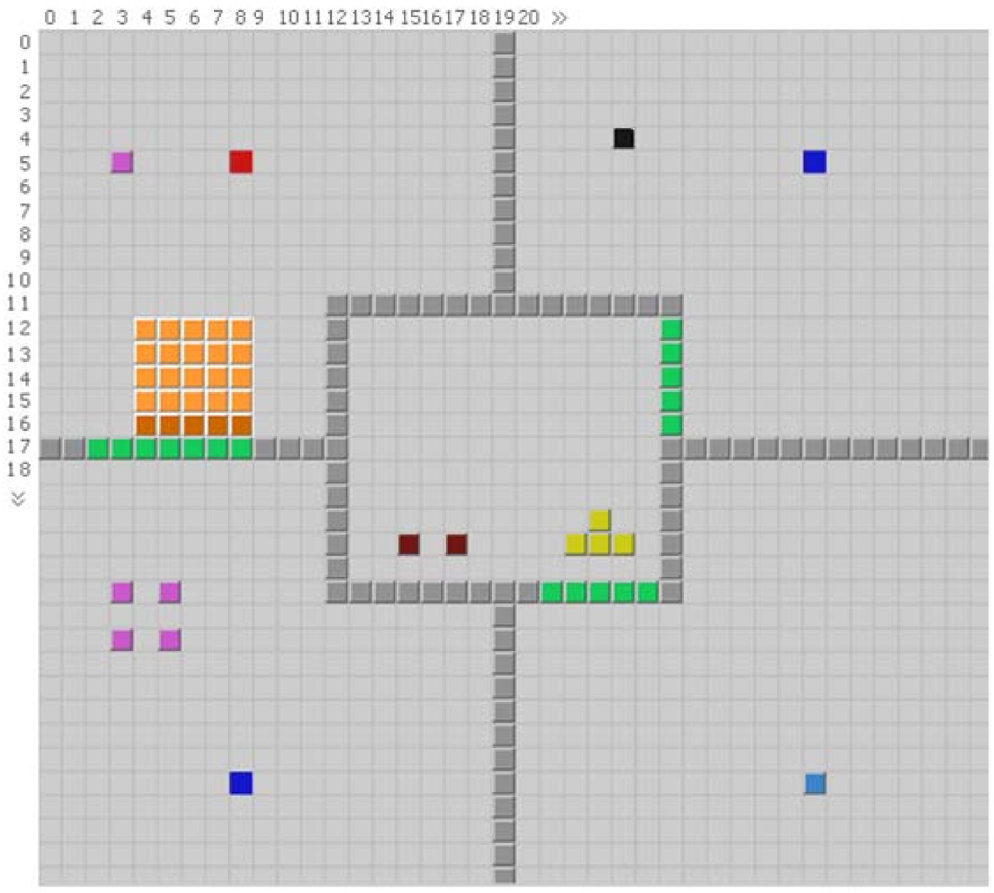


The Robot can use its tractor beam to pickup packages in an area of 5x5 units in front of its face. The Robot can pickup only one package at a time and the location of a package is specified in a coordinate system identical to that of the pictures. To pickup a package, there has to be enough space available in the Robot's package store for the whole package.

The Robot can also use its tractor beam to drop packages in its store to an area of 5x5 units in front of its face. The Robot can drop only one package at a time and the location to drop the package is specified in a coordinate system identical to that of pictures. To drop a package, the drop location has to be empty.

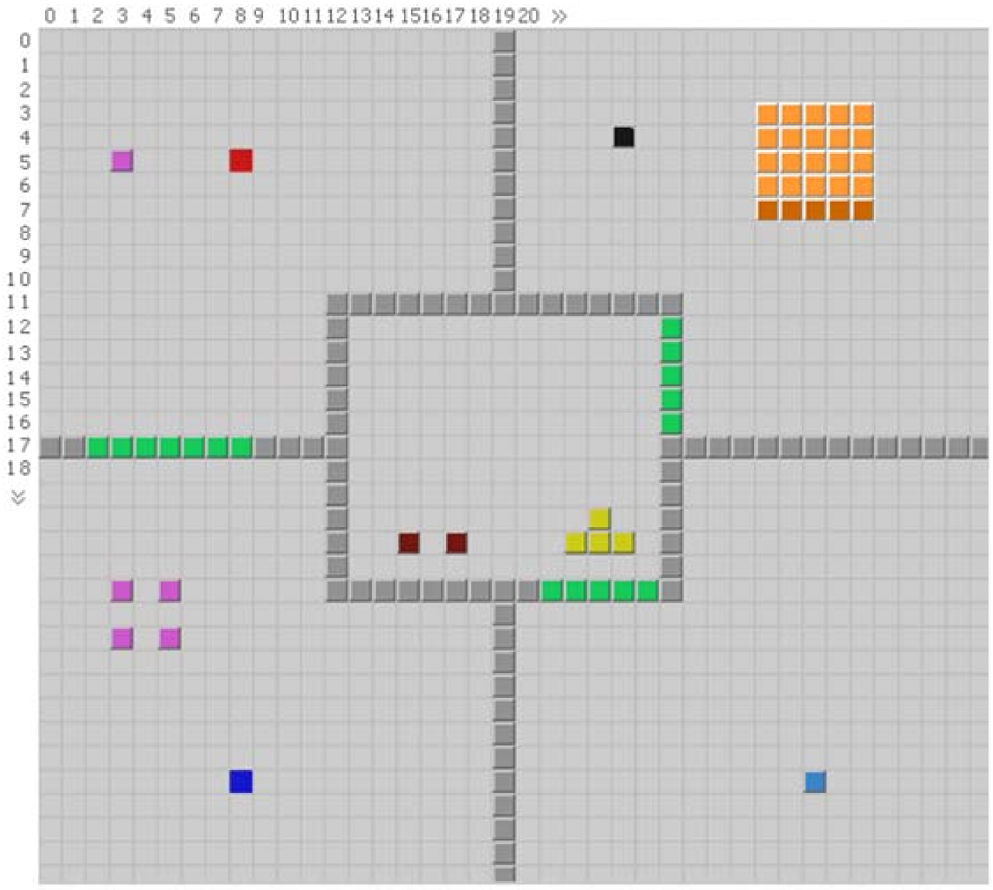
The Robot can open doors. To open a door the Robot must place itself right next to and facing the door. The Robot must also be at the left end of the door. Once open, the doors do not close.

The diagram below show a Robot just before opening a door.

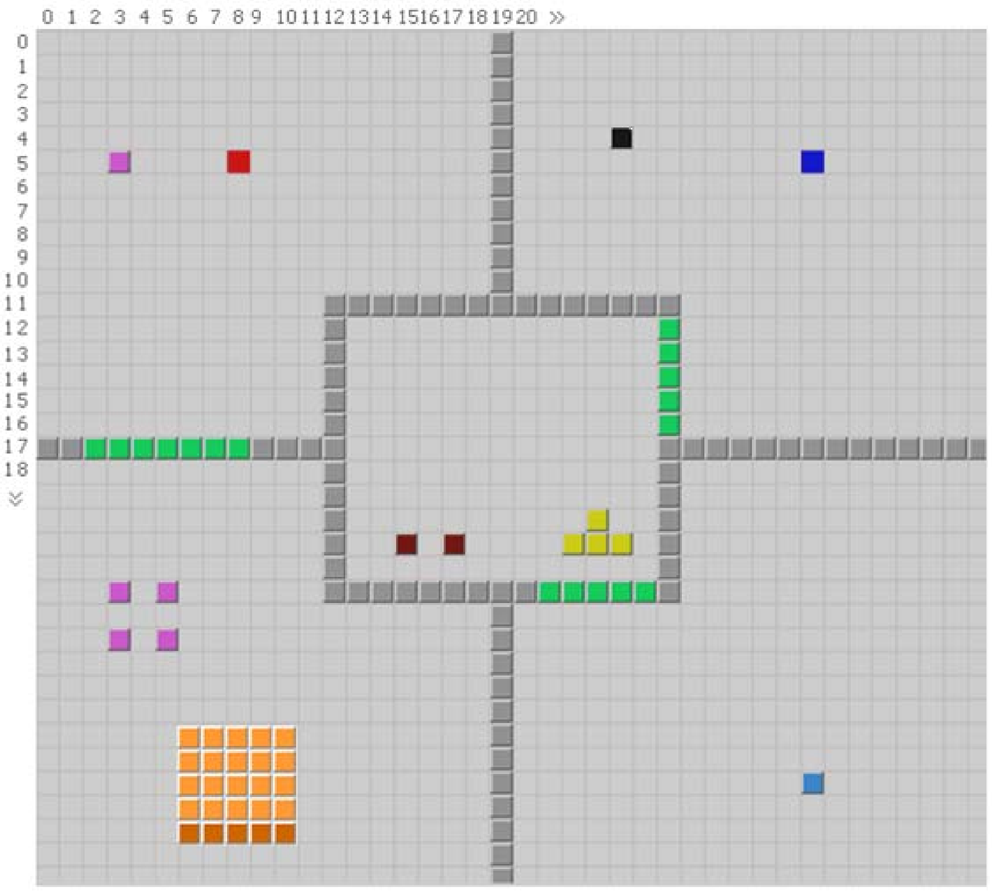


Teleport Beams operate in pairs and are used to travel between two locations in the spaceship with minimal use of battery power. A Teleport Beam is 1 unit wide, 1 unit long, and 5 units high. The Teleport Beams are scattered throughout the spaceship and can be identified through pictures. To teleport, the Robot must center itself on the Teleport Beam. After teleportation, the Robot would be centered on the destination teleport beam and would be facing the same direction as the direction in which it was facing before the teleportation. If a destination teleport area is blocked by packages, the Robot cannot teleport.

The diagram below show a Robot just before teleporting.



The diagram below show the Robot just after teleporting.



Requirement

To win the competition, you must write, using the API provided, a robot (an executable program) that explores the spaceship, picks up the valuable packages and returns to the starting location. This must be performed without damaging the robot and before the robot's battery runs out.

Package Color	Points
Yellow	20
Pink	15
Brown	10
Black	0

The Robot starts with 100% health and will lose 1% of health every time it collides with a wall, a door, or a package. It will also lose 1% health if it picks up a black package. The height of black packages has no effect on the loss of health.

The following table contains the points earned for each type of package one unit high. The points should be multiplied by the height of the package to obtain the actual points earned for picking up the package. Note that the Robot has limited space to store packages.

Black packages are toxic and picking them up earns you no points and causes the Robot to lose health.In addition to packages, battery packs can be picked up to increase the Robot's battery life provided that it does not exceed the maximum battery life of the Robot. Picking up a battery pack one unit high would cause the Robot to increase its battery life by 100 battery units. This value should be multiplied by the height of the battery pack to obtain the actual battery points.

The Robot has a carrying capacity of 120 square units excluding battery packs.

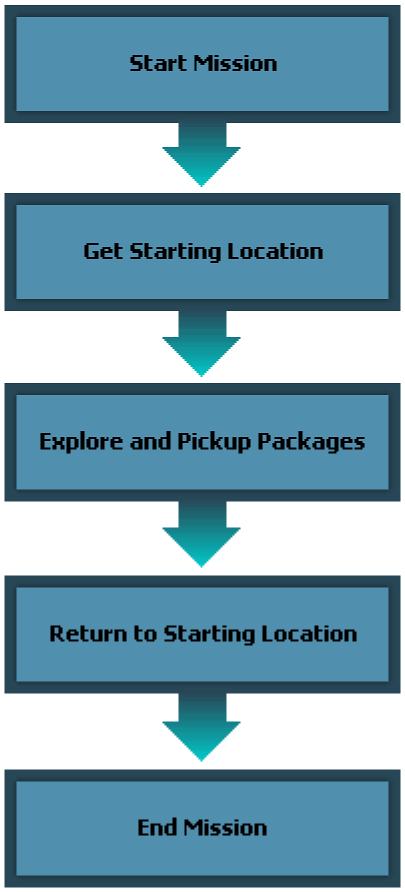
Battery packs are not added to the storage. They replace used up battery packs.

All actions of the Robot consume battery. The Robot would consume one battery unit to process an instruction to perform an action. This unit would be consumed even if the action was unsuccessful. The actions would consume varying amount of battery units depending on the amount of work the Robot has to perform. In addition, the Robot will lose battery power when idle. Idle time is defined as the time between API calls. The Robot will lose 50 battery units for every idle second. The following table contains the battery units consumed for each action.

Action	**Battery Units**	**Details**
Turn	2	The Robot consumes the same battery units to turn 90 or 180 degrees.
Move	10	10 is the battery units required to move the robot distance of one unit without any packages onboard. If the Robot is carrying packages, total battery consumption can be computed as follows: 1(?) + 10 + (Total Height of Packages) * 2 Note that, for example, if the Robot is instructed to move 5 units and it has a collision after moving 3 units, the movement would stop, and battery consumption would be for moving 3 units.
Take Picture	2	
Pickup Package	3	The Robot consumes the same amount of battery to pickup a package from any location in the 5x5 area in front of the Robot. The height of the package being picked up has no effect on the battery consumed.
Drop Package	2	The Robot consumes the same amount of battery to drop a package to any location in the 5x5 area in front of the Robot. The height of the package being dropped has no effect on the battery consumed.
Open Door	0	Size of the door has no effect on the battery consumed.
Teleport	0	Number of packages the Robot is carrying has no effect on the battery consumed.
Get Location	0	
Get Info	0	This is a utility action which can be used to retrieve the Robot's health, available battery, and the current score. The available battery would be transient value since battery is reduced for idle time.

Program Structure and Restrictions

The general structure of your Robot program should follow is shown in the diagram below.



Your program should call the StartMission() method of the API as soon as possible. If your application fails to call StartMission() method within three seconds of starting, the Robot will be disqualified when we run our batch evaluation process.

After returning from the StartMission() method, the Robot would be placed at the starting location of the current layout of the spaceship loaded to *Spaceship.exe*. The Robot would be centered on the Starting Location. Your application should retrieve this location using GetLocation() method of the API and remember it for later use.

Next, you should implement your algorithm to achieve the competition objectives by using Turn(), Move(), TakePicture(), Pickup(), Drop(), OpenDoor(), Teleport(), and GetInfo() methods of the API. The GetInfo() method can be used retrieve the remaining battery units. However this value would be transient as battery power would be lost when idling.

Finally the Robot should return to the starting location and call the EndMission() method of the API. Returning to the starting location is very important as if the EndMission() method is called from any other location, the Robot would be disqualified. Note the Robot must be centered on the Starting Location. After calling EndMission() the application should terminate.

Your Robot can be a console application or a GUI but it should not require any human interaction. That is, your program should not prompt a user for any input. The reason for this restriction is that all Robots are evaluated in an automated batch process. Your application would be run on a Pentium 4 with 256 MB of memory. Excessive memory use should be avoided. Your application should not create log files or attempt to write any information to the host machine. The reason for this restriction is that we would be running a large number of Robots on the same machine and it would be very time consuming to clean after each Robot.

Disqualifications

The Robot would be disqualified if any of the following situations occur.

- The Robot runs out of battery before ending the mission.
- The Robot loses all its health before ending the mission.
- The Robot terminates mission without returning to the starting location.
- The Robot terminates without ending the mission.
- The Robot fails to start the mission within three seconds of starting the program.
- Your program attempts to harm the machine on which it runs.

SDK

The SDK is available on the following operating systems.

- Windows XP
- Windows 2000
- Windows NT

The API is available in following programming languages.

- C++
- Java
- Visual Basic
- .NET

The following table contains the supported compilers for the above languages. If you don't own the required compilers please see our website for an alternative set of libraries.

Language	Compiler
C, C++	Microsoft Visual C++ 6 or 2003
Java	JDK 1.4 or above
Visual Basic	Microsoft Visual Basic 6
.NET	Microsoft .NET framework 1.x

The following table contains all executable available with the SDK.

Executable	Description
Spaceship.exe	Creates a virtual ship to test your Robot.
ShipBuilder.exe	Creates spaceships with different layouts which can be loaded on to *Spaceship.exe*.
ManCtrl.exe	This GUI can be used to manually control the Robot. It is written using the C API to demonstrate the behavior of API calls.

The Following documentation is available with the SDK.

- This document.
- MARC Rules.
- User manuals for *Spaceship.exe* and *ShipBuilder.exe*
- API Reference manuals for each of the programming languages.

Note that documents except the Rules are provided in HTML Help file format (.chm) for easer reference.

In addition all the link and dynamic libraries necessary for different programming languages and sample maps and skins can be found in the installation directory.

Testing the Robot

To test the Robot, run *Spaceship.exe*, load a layout of a spaceship, and start your Robot. Scores and actions of the Robot can be seen in 2D or 3D on the *Spaceship.exe* application.

You should test your Robot on a variety of spaceship layouts as the competition layouts are not published. You can use simple maps provided with the installation, download sample maps from our website, or create your own using the *ShipBuilder.exe* application.

The API you used to create the Robot would use TCP/IP to communicate with the *Spaceship.exe*. The TCP/IP port number can be set from *Spaceship.exe* and may have to be unblocked if you are using a firewall.

Note that you can test only one instance of a Robot at any given time.

Submitting the Robot

Once you are satisfied that your Robot meets all the requirements of the competition, you can upload your Robot to our site by using a web browser or *Spacelab.exe*. All Robots are evaluated on each Friday and scores are posted on the website. You may submit the Robot any number of times. We will only retain the latest copy. Winners are those with the highest average points at the end of the competition

Please confirm that we have your latest version by verifying that the upload time and the checksum displayed at our website is correct. We recommend that you don't wait till the last day of the competition to upload your Robot as the site may get overloaded. .

[1] Java is a trademark or registered trademark of Sun Microsystems, Inc. in the U.S. or other countries.
[2] Windows is a registered trademark of Microsoft Corporation in the United States and other countries.