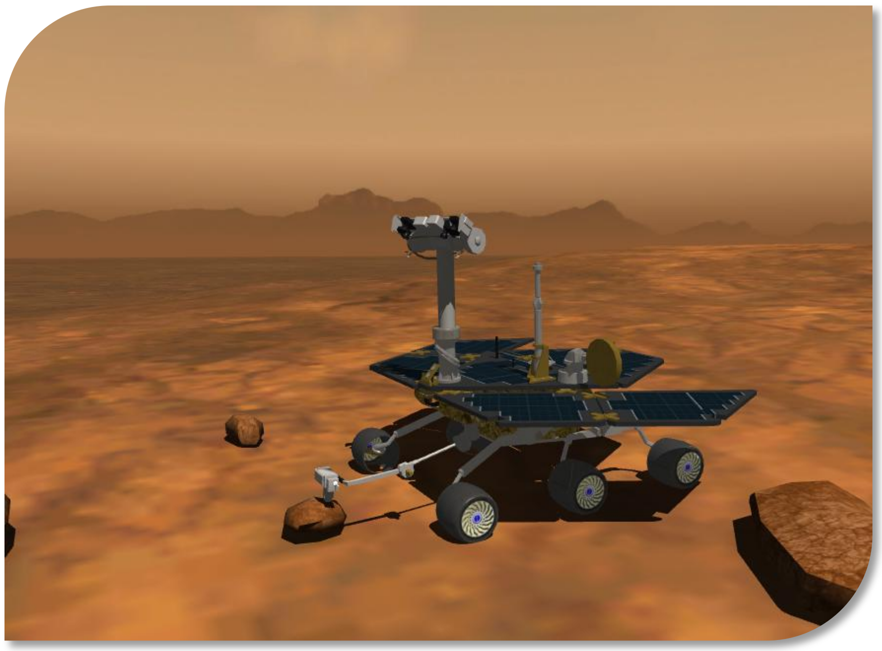
ROBOCHAMPS MARS CHALLENGE



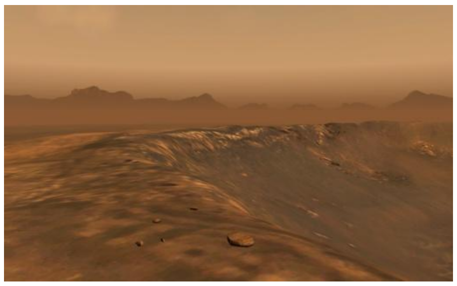
Congratulations, you have just landed on Mars, but your mission is far from being over! You must now explore the area with your own copy of the NASA Rover robot. You will have to drive the Rover all around a crater and use its special tools to analyze the surface and characterize a wide range of rocks, soils, and dust. Remember that the scientific goal of the mission is to find specimens that hold clues to past water activity on the red planet…

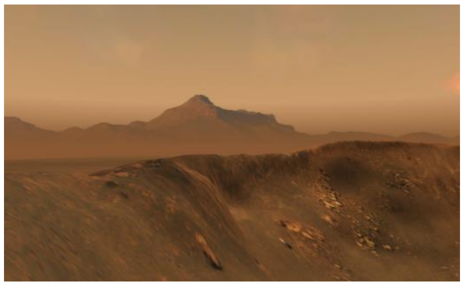
CONTENTS

CHALLENGE RULES

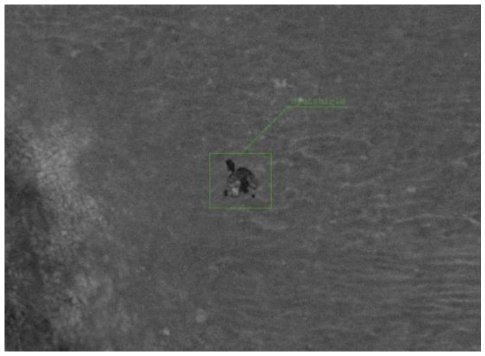
GENERAL RULES

The mission goal is to collect data around the Endurance crater. We have selected six positions that could be favorable for collecting. Your job is to program the rover to navigate through the areas and collect information with its spectrometer. For selected positions, you will be awarded 50 points if you reach the area and 50 points if you analyze a rock in the area.





The heat shield that has protected the Lander during the descent should be near the last selected positions. It would be interesting if you find it and go close enough to see how it deals with the damages. You will be awarded 50 points if you find and approach the heat shield. Here a picture of the shield taken by a satellite.



A special 75 points bonus will be given at the end of your mission if you have visited, in order, all the selected positions and the heat shield.

To accomplish this mission, you can use all rover sensors and actuators described below. Note that you can send an "End Mission" message to Earth whenever you want. You could submit your score if the mission is partially succeeded. The maximum score you could reach is 725 points: 6 * 50 (area visited) + 6 * 50 (rocks analyzed) + 50 (heat shield) + 75 (navigation bonus).

SENSORS AND ACTUATORS

CAMERAS

The Mars rover has four "eyes". A panoramic camera and a navigation camera mounted on the head, and two hazard avoidance cameras mounted on the lower portion of front and rear of the rover.

Cameras	Mode	Resolution	FOV
Panoramic camera	Color	128x128	16°
Navigation camera	Color	128x128	45°
Front camera	Black and white	128x128	120°
Rear camera	Black and white	128x128	120°

SPECTROMETER

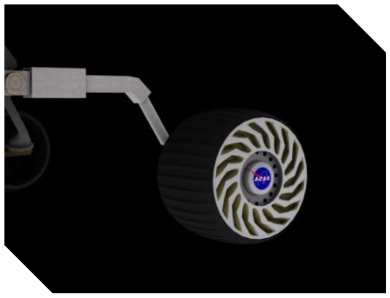
The spectrometer is mounted on the turret at the end of the arm. It analyses rocks when it's in contact. The MSRDS service sends notification when the spectrometer state changes. Here, the state fields description:

Name	Type	Description
SampleId	Integer	Unique id for each area. If the spectrometer analyzes rocks from the same area, this id will not change.
InContact	Boolean	True when the spectrometer is in contact with a rock, false otherwise.
Surface	Array of float	Analysis result (arbitrary values).
Brushed	Array of float	Analysis result (arbitrary values).
PostRAT	Array of float	Analysis result (arbitrary values).

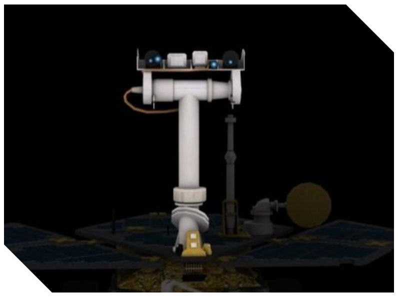
DRIVE

The Mars rover has six wheels commanded by a differential drive. To control it, you have to send left and right power to the MSRDS service. Power value varies between -1 and 1. It controls the rotation speed and direction of each wheel.

But beware! Mars soil is uneven and composed of dust. The wheels could slip and you have to use other sensors to make sure that you move in the right direction.



HEAD

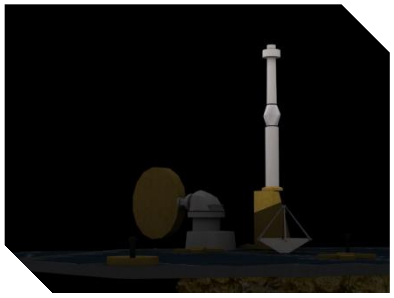


The rover head can rotate horizontally and vertically. It's controlled by a "Pan Tilt" MSRDS service. It accepts several operations to control both these rotations. As panoramic and navigation cameras are mounted on the head, move it to choose the direction you want to look at.

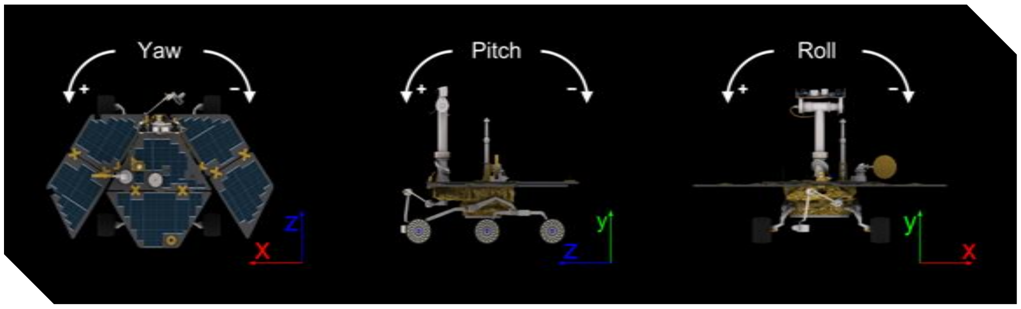
COMLINK

The comlink informs the rover on its currents positions and orientations. It also provides information on the special areas and it can send messages to Earth. The MSRDS service sends periodical notifications with its state. Here, the state fields descriptions:

Name	Type	Description
Timestamp	Date	ComLink state is time stamped.
Yaw	Float	The rover yaw in degree [-180°, 180°]
Pitch	Float	The rover pitch in degree [-180°, 180°]
Roll	Float	The rover roll in degree [-180°, 180°]
Position	Coordinates	Coordinates of the rover in three dimensions (x, y, z).
CurrentArea	Area	Current area if the rover is in a special position. An area is defined by a name and a position.
Areas	Array of Area	Full list of known areas.

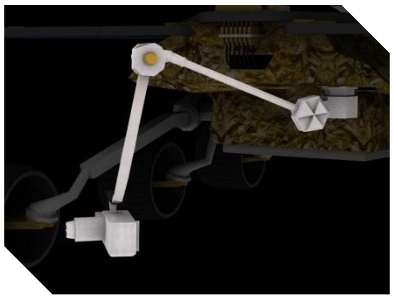


Three dimensions coordinate are given in a classic right handed coordinate system, Y on top.



When the rover thinks that its task is complete, it MUST send an “End Mission” message to Earth. ComLink MSRDS service port accept end mission message.

ARTICULATED ARM



The rover arm has three joint like a human arm: a shoulder, an elbow and a wrist. These joints are controlled by five motors that allow the followings movements:

Joint	Movement
Shoulder	Two motors control the shoulder. One for horizontal movements through 160°, one for vertical movements through 70°.
Elbow	The elbow is controlled by one motor. It can move through 290°.
Wrist	The wrist can move vertically through 340° and can spin horizontally through 350°.

The arm is managed by a MSRDS service that implement generic articulated arm contract. Note that the inverse kinematic methods (get and set end effector pose) are not implemented, you have to move the joints one by one. Joint names are: ShoulderH, ShoulderV, Elbow, WristH and WristV.

CHALLENGE PACKAGE

MINIMAL REQUIREMENT

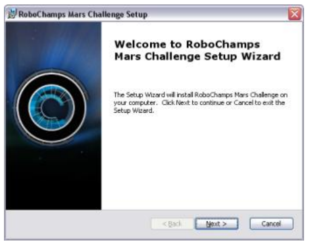
HARDWARE

- 1Ghz processor or greater
- 512 MB of memory or greater
- Graphics card supporting DirectX 9.0c (or later) and Shader Model 2.0+ with 64MB of video memory or greater (ATI Radeon 9800+ or NVIDIA FX series or later)
  Recommended: Graphics card supporting DirectX 9.0c (or later) and Shader Model 3.0+ with 128MB of video memory or greater (ATI Radeon x1300 or NVIDIA 6 series or later)
- An Internet connection

SOFTWARE

- Microsoft Robotics Developer Studio 2008 Express Edition (works also with Standard and Academic editions)
  http://www.microsoft.com/downloads/details.aspx?FamilyID=84c5b49f-0f9c-4182-a267-a951328d3fbd&displaylang=en
- Microsoft Visual C# 2008 Express Edition (or greater)
  http://www.microsoft.com/express/vcsharp/

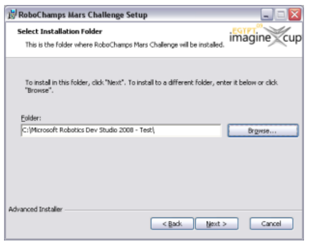
INSTALLATION


RoboChamps Mars Challenge Setup

Welcome to RoboChamps Mars Challenge Setup Wizard

The Setup Wizard will install RoboChamps Mars Challenge on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

When you double click on the installation file, the welcome screen should appear.


RoboChamps Mars Challenge Setup

Select Installation Folder

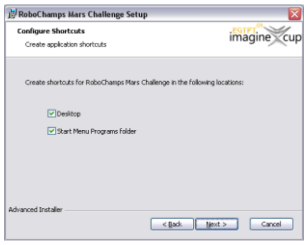
This is the folder where RoboChamps Mars Challenge will be installed.

To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder: C:\Microsoft Robotics Dev Studio 2008 - Test\

Advanced Installer

Choose the installation folder. **It MUST be the same that the MSRDS installation directory**. For example, if you have installed MSRDS in C:\Users\Me\Microsoft Robotics Dev Studio 2008 Express, you MUST choose this folder.


RoboChamps Mars Challenge Setup

Configure Shortcuts

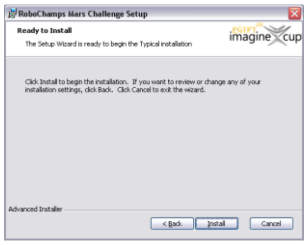
Create application shortcuts

Create shortcuts for RoboChamps Mars Challenge in the following locations:

- [✓] Desktop
- [✓] Start Menu Programs folder

Advanced Installer

Select where you want the setup package to create shortcuts.


RoboChamps Mars Challenge Setup

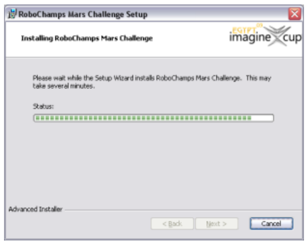
Ready to Install

The Setup Wizard is ready to begin the Typical installation

Click Install to begin the installation. If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

Advanced Installer

Normally, the installer should be ready to install, just confirm the installation.


RoboChamps Mars Challenge Setup

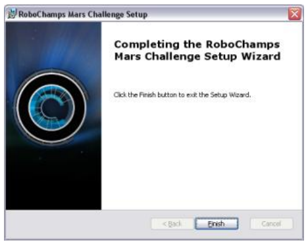
Installing RoboChamps Mars Challenge

Please wait while the Setup Wizard installs RoboChamps Mars Challenge. This may take several minutes.

Status:

Advanced Installer

The installation should be quite fast, just wait.


RoboChamps Mars Challenge Setup

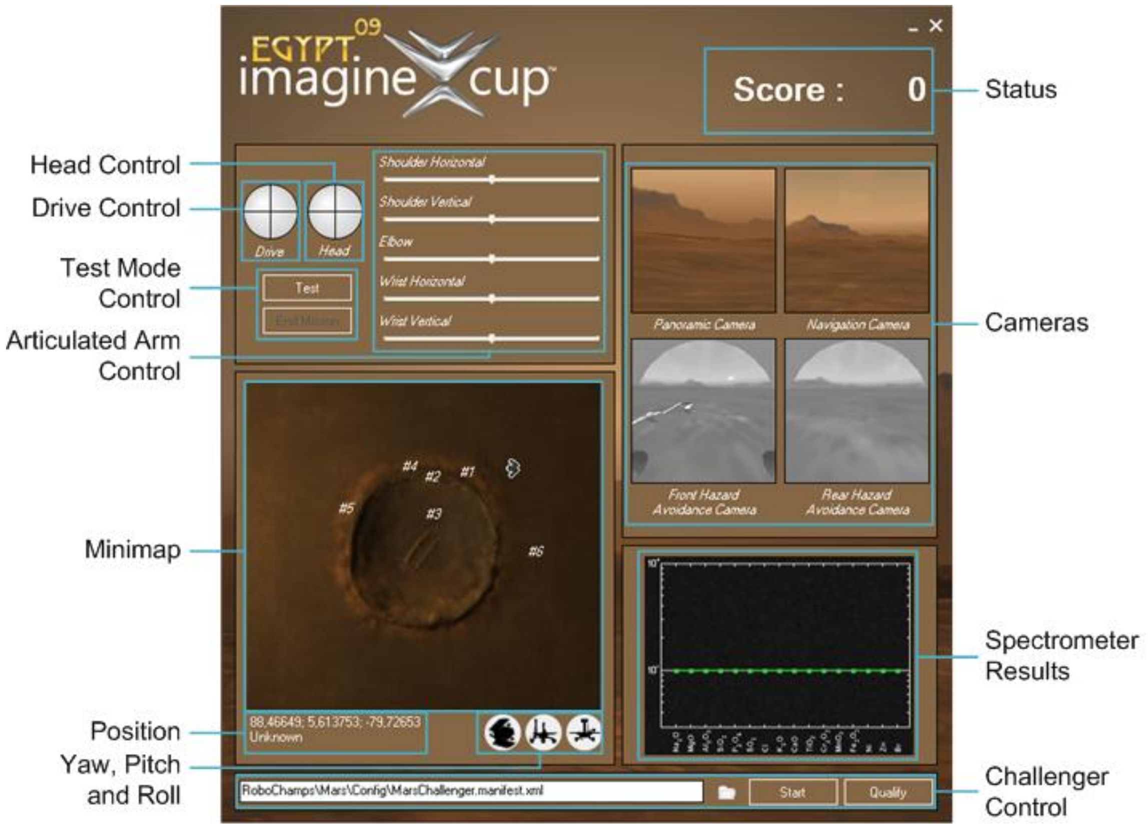
Completing the RoboChamps Mars Challenge Setup Wizard

Click the Finish button to exit the Setup Wizard.

If you see this screen, you are ready to drive on Mars!

INTERFACE

QUICK REFERENCE



Status	Shows your current score and displays status messages
Test Mode Control	Allows you to launch test session. Click on “Test” to launch a session and on “End Mission” to finish the session
Head Control	Controls the rover’s head in test mode
Drive Control	Controls the rover’s drive in test mode
Articulated Arm Control	Controls the rover’s arm in test mode
Cameras	Displays current cameras frames
Minimap	Shows a minimap area with information from ComLink (areas and rover position)
Postion	Shows numeric values of rover position and the current area
Yaw, Pitch and Roll	Indicates the yaw, pitch and roll of the rover
Spectrometer Results	Displays current spectrometer results
Challenger Control	Launches a challenger manifest in normal or qualify mode

Don’t forget that you can choose to view a specific camera in main simulation window, by using the “Camera” menu. An additional pursuit camera is available.

COMMON USES

Free mode

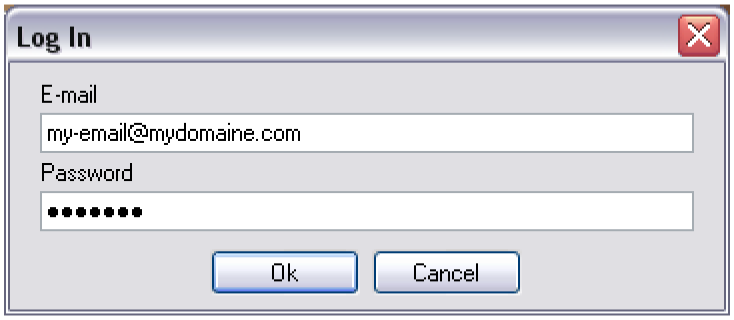
To test the rover actuators and sensors, click on “Test” button. A new mission is launched with rover controls enabled. You can drive around, move the arm and the head and view sensors data. The referee is running too and gives you your current score. When you are done, just click on “End Mission” to finish test session.

Test your challenger

To test your challenger, choose your manifest and click on “Start” button. By default, the sample challenger manifest is selected. Your robot must send “End Mission” message to the comlink to finish the session.

Qualify yourself

When you think your challenger is good enough, select your manifest and click on “Qualify” button. A message box should appear allowing you to log on Imagine Cup server.



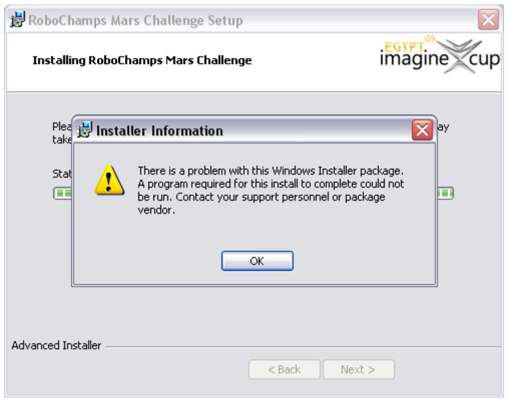
my-email@mydomaine.com

●●●●●●●

If your login is correct, your challenger will start his mission just as it would have done in the test mode. When the challenger sends the “End Mission” message to the comlink, your score is calculated and sent to the server. You should then be able to see your ranking on Imagine Cup leader board.

TROUBLESHOOTING

Installation fails


RoboChamps Mars Challenge Setup
Installing RoboChamps Mars Challenge
Installer Information
OK
Advanced Installer
< Back
Next >

There is a problem with this Windows Installer package. A program required for this install to complete could not be run. Contact your support personnel or package vendor.

If you see this screen, verify that you have chosen the right installation folder. The installation folder must be the same as the one where MSRDS is installed. If the folder is correct, something must be wrong with your MSRDS installation, you should try to reinstall it.

Submit score fails or you can’t login

A web service is used to submit score. If the submission fails or you can’t login, check you Internet connection. If you have a firewall, check that DssHost32.exe is allowed to communicate with a distant server on port TCP 80.

If you are behind an authentication web proxy, you can provide your proxy credentials in the service client configuration (view “Configure service client” in “Advanced Topics”).

Simulation window doesn’t show up

Check that your graphic adapter follows minimum requirements given above and that you have installed the latest version of your video card drivers.

SAMPLE CHALLENGER SERVICE

CODE EXPLANATIONS

The purpose of this sample is to give an example of a service oriented structure. It doesn't implement a rover "AI', as you'll have to write your own, based on the sample structure or with you own architecture.

The sample challenger is divided in five services: four image processor services (one per camera) and one main service, Mars Challenger.

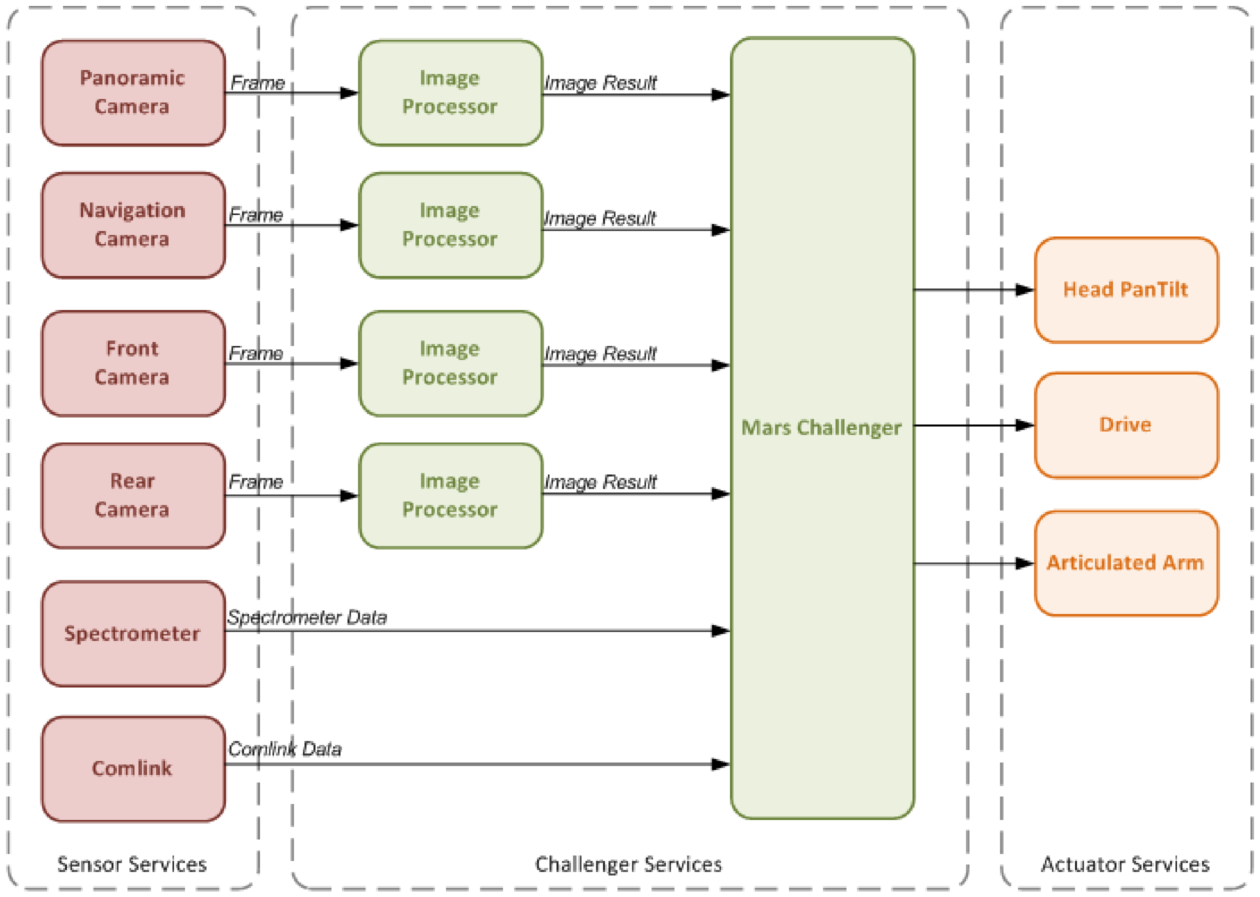


IMAGE PROCESSOR

The image processor service analyzes camera frames. It takes a camera as partner, subscribes to its notifications, analyzes each frame and sends notifications with the analysis results. The image processing is performed by a "processor", a simple object with a processing method that could be different for each instance of the service (a processor is associated with the service name).

Four instances are launched by the challenger manifest: one for the panoramic camera that uses `PanCamProcessor`, one for the navigation camera that uses `NavCamProcessor` and two for the front and rear cameras that uses `BasicCamProcessor`.

Let's take a look at a processor code (from ImageProcessor.cs).

```
internal class NavCamProcessor : IImageProcessor
{
    public ImageResult Process(QueryFrameResponse image)
    {
        int offset = 0;

        for (int y = 0; y < image.Size.Height; ++y)
        {
            offset = y * image.Size.Width * 3;

            for (int x = 0; x < image.Size.Width; ++x)
            {
                int r, g, b;

                b = image.Frame[offset++];
                g = image.Frame[offset++];
                r = image.Frame[offset++];

                // Do some stuff
            }
        }

        // Set what you want in the image result
        // and post it on result port
        ImageResult result = new ImageResult();
        result.Timestamp = DateTime.Now;
        return result;
    }

}
```

A processor is a simple class that implements `IImageProcessor` interface. The Process method takes a query frame response (frame bytes + image information) and must return an `ImageResult` object. This class is defined in ImageProcessorTypes.cs.

```
[DataContract]
public class ImageResult
{
    [DataMember]
    public DateTime Timestamp { get; set; }

    // You could add what you want in image result

}
```

The purpose of this object is to carry processing result through service oriented architecture. It's common for all processor. If you want, for example, find a pixel position in the image, you should add coordinate in `ImageResult` class:

```
[DataMember]
public int X { get; set; }
[DataMember]
public int Y { get; set; }
```

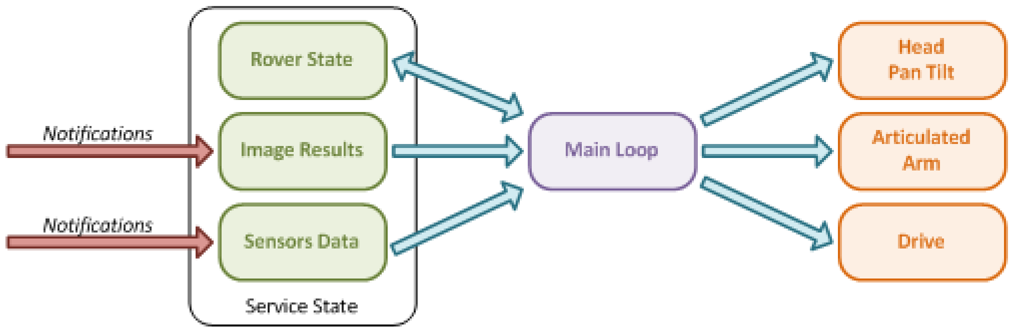
and assign it in the Process method:

```
ImageResult result = new ImageResult();
result.Timestamp = DateTime.Now;
result.X = calculatedX;
result.Y = calculatedY;
return result;
```

These fields will be accessible in the challenger main service.

CHALLENGER

Mars challenger service is the main service. It sends command to rover actuators depending on notifications from sensors and from image processors.



Sensors and image processors notification handlers update the service state with refreshed data.

```
…
private void SpectroReplaceHandler(spectro.Replace replace)
{
    _state.SpectrometerData = replace.Body;
}

private void PanCamProcessImageHandler(processor.ProcessImage process)
{
    _state.PanCamData = process.Body;

}

…
```

State contains also a “Rover State” that indicates the current status of the rover. Only three states are declared, you should add your own.

```
[DataContract]
public enum RoverState
{
     Initialize,
     Navigate,
     EndMission,
}
```

The key method of this service is `MainLoopHandler`. This method sends commands to actuators according the rover state and the sensors data. It can be called periodically or asynchronously, when the service state is refreshed.

We provide a helper for calling this loop after a given amount of time, `RearmMainLoop`.

In the sample, this function performs basic operations. Arm and head are moved in the “Initialize” state, rover move ten meters in “Navigate” state and End Mission signal is sent in “EndMission” state. You should improve this method to give intelligence to your rover, the only purpose of given one is to demonstrate how to interact with sensors and actuators.

DEBUGGING

You can add debug messages to your mars challenger service that will help you track what it is doing. Look at the LogInfo() calls as an example.

```
LogInfo(LogGroups.Console, String.Format(message, args));
```

Log messages should print time stamped in the console window.

When you are testing your final service, you should minimize the amount of debug messages that are printed; it has a significant impact on the service performance.

ADVANCED TOPICS

PLAYER MANIFEST

Player selection is made by selecting a manifest file. If you are familiar with Microsoft Robotics Developper Studio, you should be familiar with this kind of file. It lists services to create with their partnerships. When you select a manifest file in the challenger selection interface, the referee analyses the partnerships and sets “dssp:Service” element with the right exisiting.

Here is the sample mars challenger manifest. Note that “dssp:Service” elements are filled with special keywords that will be replace with right services addresses. Special keywords are: PanCam, NavCam, FrontCam, RearCam, ComLink, Spectro, Head, Drive and Arm.

```
<?xml version="1.0"?>
<Manifest xmlns:challenger="http://www.robochamps.com/2009/02/marschallenger.html"
          xmlns:processor="http://www.robochamps.com/2009/02/marschallenger/imageprocessing.html"
          xmlns:this="urn:uuid:248a1140-46fe-4778-b2fc-9e33e6a69e3c"
          xmlns:dssp="http://schemas.microsoft.com/xw/2004/10/dssp.html"
          xmlns="http://schemas.microsoft.com/xw/2004/10/manifest.html">
  <CreateServiceList>

    <!-- Camera Processors -->

    <ServiceRecordType>
      <dssp:Contract>http://www.robochamps.com/2009/02/marschallenger/imageprocessing.html</dssp:Contract>
      <dssp:Service>http://localhost/pancamprocessor</dssp:Service>
      <dssp:PartnerList>
        <dssp:Partner>
          <dssp:Contract>http://schemas.microsoft.com/robotics/2006/05/webcamservice.html</dssp:Contract>
          <dssp:Service>PanCam</dssp:Service>
          <dssp:Name>processor:Camera</dssp:Name>
        </dssp:Partner>
        </dssp:PartnerList>
      <Name>this:PanCamProcessor</Name>
    </ServiceRecordType>

    <ServiceRecordType>
      <dssp:Contract>http://www.robochamps.com/2009/02/marschallenger/imageprocessing.html</dssp:Contract>
      <dssp:Service>http://localhost/navcamprocessor</dssp:Service>
      <dssp:PartnerList>
        <dssp:Partner>
          <dssp:Contract>http://schemas.microsoft.com/robotics/2006/05/webcamservice.html</dssp:Contract>
          <dssp:Service>NavCam</dssp:Service>
          <dssp:Name>processor:Camera</dssp:Name>
        </dssp:Partner>
      </dssp:PartnerList>
      <Name>this:NavCamProcessor</Name>
    </ServiceRecordType>

    <ServiceRecordType>
      <dssp:Contract>http://www.robochamps.com/2009/02/marschallenger/imageprocessing.html</dssp:Contract>
      <dssp:Service>http://localhost/frontcamprocessor</dssp:Service>
      <dssp:PartnerList>
        <dssp:Partner>
          <dssp:Contract>http://schemas.microsoft.com/robotics/2006/05/webcamservice.html</dssp:Contract>
          <dssp:Service>FrontCam</dssp:Service>
          <dssp:Name>processor:Camera</dssp:Name>
        </dssp:Partner>
      </dssp:PartnerList>
      <Name>this:FrontCamProcessor</Name>
    </ServiceRecordType>

    <ServiceRecordType>
      <dssp:Contract>http://www.robochamps.com/2009/02/marschallenger/imageprocessing.html</dssp:Contract>
      <dssp:Service>http://localhost/rearcamprocessor</dssp:Service>
      <dssp:PartnerList>
        <dssp:Partner>
          <dssp:Contract>http://schemas.microsoft.com/robotics/2006/05/webcamservice.html</dssp:Contract>
          <dssp:Service>RearCam</dssp:Service>
          <dssp:Name>processor:Camera</dssp:Name>
        </dssp:Partner>
      </dssp:PartnerList>
```

```xml
      <Name>this:RearCamProcessor</Name>
    </ServiceRecordType>

    <!-- Challenger -->

    <ServiceRecordType>
      <dssp:Contract>http://www.robochamps.com/2009/02/marschallenger.html</dssp:Contract>
      <dssp:PartnerList>
        <dssp:Partner>
          <dssp:Contract>http://www.robochamps.com/sensors/2009/02/marscomlink.html</dssp:Contract>
          <dssp:Service>ComLink</dssp:Service>
          <dssp:Name>challenger:ComLink</dssp:Name>
        </dssp:Partner>
        <dssp:Partner>
          <dssp:Contract>http://www.robochamps.com/sensors/2009/02/marsspectrometer.html</dssp:Contract>
          <dssp:Service>Spectro</dssp:Service>
          <dssp:Name>challenger:Spectro</dssp:Name>
        </dssp:Partner>
        <dssp:Partner>
          <dssp:Contract>http://www.robochamps.com/2009/02/marschallenger/imageprocessing.html</dssp:Contract>
          <dssp:PartnerList />
          <dssp:Name>challenger:PanCamProcessor</dssp:Name>
          <dssp:ServiceName>this:PanCamProcessor</dssp:ServiceName>
        </dssp:Partner>
        <dssp:Partner>
          <dssp:Contract>http://www.robochamps.com/2009/02/marschallenger/imageprocessing.html</dssp:Contract>
          <dssp:PartnerList />
          <dssp:Name>challenger:NavCamProcessor</dssp:Name>
          <dssp:ServiceName>this:NavCamProcessor</dssp:ServiceName>
        </dssp:Partner>
        <dssp:Partner>
          <dssp:Contract>http://www.robochamps.com/2009/02/marschallenger/imageprocessing.html</dssp:Contract>
          <dssp:PartnerList />
          <dssp:Name>challenger:FrontCamProcessor</dssp:Name>
          <dssp:ServiceName>this:FrontCamProcessor</dssp:ServiceName>
        </dssp:Partner>
        <dssp:Partner>
          <dssp:Contract>http://www.robochamps.com/2009/02/marschallenger/imageprocessing.html</dssp:Contract>
          <dssp:PartnerList />
          <dssp:Name>challenger:RearCamProcessor</dssp:Name>
          <dssp:ServiceName>this:RearCamProcessor</dssp:ServiceName>
        </dssp:Partner>
        <dssp:Partner>
          <dssp:Contract>http://www.simplysim.net/robotics/actuators/2009/02/pantilt.html</dssp:Contract>
          <dssp:Service>Head</dssp:Service>
          <dssp:Name>challenger:Head</dssp:Name>
        </dssp:Partner>
        <dssp:Partner>
          <dssp:Contract>http://schemas.microsoft.com/robotics/2006/05/drive.html</dssp:Contract>
          <dssp:Service>Drive</dssp:Service>
          <dssp:Name>challenger:Drive</dssp:Name>
        </dssp:Partner>
        <dssp:Partner>
          <dssp:Contract>http://schemas.microsoft.com/2006/06/articulatedarm.html</dssp:Contract>
          <dssp:Service>Arm</dssp:Service>
          <dssp:Name>challenger:Arm</dssp:Name>
        </dssp:Partner>
      </dssp:PartnerList>
      <Name>this:Challenger</Name>
    </ServiceRecordType>
  </CreateServiceList>

</Manifest>
```

Any valid manifest file should work and will be launched by considering the rules explained above.

CONFIGURE SERVICE CLIENT

Web service client configuration is defined in Microsoft.Robotics.RoboChamps.ServiceClient.Y2009.M02.dll.config file under MSRDS\bin folder.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
    <appSettings>
      <!--
      Use these fields if your are behind an HTTP web proxy.
      Domain is optional.
      Proxy address and port MUST be configured in Internet Settings (IE Settings).
      -->
      <!--
      <add key="ProxyUsername" value=""/>
      <add key="ProxyPassword" value=""/>
      <add key="ProxyDomain" value=""/>
      -->
    </appSettings>
    <system.serviceModel>
        <behaviors />
        <bindings>
            <basicHttpBinding>
                <binding name="AuthServiceSoap" closeTimeout="00:01:00" openTimeout="00:01:00"
                    receiveTimeout="00:10:00" sendTimeout="00:01:00" allowCookies="false"
                    bypassProxyOnLocal="false" hostNameComparisonMode="StrongWildcard"
                    maxBufferSize="65536" maxBufferPoolSize="524288" maxReceivedMessageSize="65536"
                    messageEncoding="Text" textEncoding="utf-8" transferMode="Buffered"
                    useDefaultWebProxy="true">
                    <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
                        maxBytesPerRead="4096" maxNameTableCharCount="16384" />
                    <security mode="TransportCredentialOnly">
                        <transport clientCredentialType="Windows" proxyCredentialType="None"
                            realm="" />
                        <message clientCredentialType="UserName" algorithmSuite="Default" />
                    </security>
                </binding>
            </basicHttpBinding>
            <customBinding>
                <binding name="AuthServiceSoap12">
                    <textMessageEncoding maxReadPoolSize="64" maxWritePoolSize="16"
                        messageVersion="Soap12" writeEncoding="utf-8">
                        <readerQuotas maxDepth="32" maxStringContentLength="8192" maxArrayLength="16384"
                            maxBytesPerRead="4096" maxNameTableCharCount="16384" />
                    </textMessageEncoding>
                    <httpTransport manualAddressing="false" maxBufferPoolSize="524288"
                        maxReceivedMessageSize="65536" allowCookies="false" authenticationScheme="Anonymous"
                        bypassProxyOnLocal="false" hostNameComparisonMode="StrongWildcard"
                        keepAliveEnabled="true" maxBufferSize="65536" proxyAuthenticationScheme="Anonymous"
                        realm="" transferMode="Buffered" unsafeConnectionNtlmAuthentication="false"
                        useDefaultWebProxy="true" />
                </binding>
            </customBinding>
        </bindings>
        <client>
            <endpoint address="http:// imaginecup.com/WebService/AuthService.asmx"
                binding="basicHttpBinding" bindingConfiguration="AuthServiceSoap"
                contract="ICAuthService.AuthServiceSoap" name="AuthServiceSoap" />
            <endpoint address="http://imaginecup.com/WebService/AuthService.asmx"
                binding="customBinding" bindingConfiguration="AuthServiceSoap12"
                contract="ICAuthService.AuthServiceSoap" name="AuthServiceSoap12" />
        </client>
    </system.serviceModel>

</configuration>
```

If you are behind a web proxy that requires an authentication, you can set your credentials under “appSettings” element (username, password and domain).

The second part of this file is a classic WCF client configuration file and should not be edited. For more documentation, read MSDN WCF section at http://msdn.microsoft.com/en-us/library/ms731354.aspx.